2

Document No.1490A-001
30 June 1990

Updated Application Blueprint Definition for C3 for the Software Technology for Adaptable, Reliable Systems (STARS) Program

Contract No. F19628-88-D-0032

Task IR20-Process/Environment Integration

CDRL Sequence No. 1490A-001

30 June 1990

Prepared for:

Electronic Systems Division
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

Prepared by:

IBM Federal Sector Division
800 North Frederick Avenue
Gaithersburg, MD 20879

90

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 30, 1990	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE Updated Application Blueprint Definition for C3	5. FUNDING NUMBERS C: F19628-88-D-0032
6. AUTHOR(S) J. Piotrowski	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) IBM Federal Sector Division 800 N. Frederick Avenue Gaithersburg, MD 20879	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Electronic Systems Division Air Force Systems Command, USAF Hanscom AFB, MA 01731-5000	10. SPONSORING/MONITORING AGENCY REPORT NUMBER CDRL Sequence No. 1490A-001
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Application blueprints serve as frameworks for designing new systems in an application domain, leading to reuse of design information and greater reuse of code. This document defines the term application blueprint, tells how to create one, and discusses the benefits and drawbacks of this approach. The appendix presents a generic specification and information about the initial domain analysis for creating an application blueprint for an air traffic control system. This paper can be the basis for future research on reusing analysis and design information.

Keywords: STARS (Software Technology for Adaptable Reliable System), (KR)

14. SUBJECT TERMS STARS, application blueprints, software reuse, air traffic control.	15. NUMBER OF PAGES 57
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

Table of Contents

List of Illustrations

Scope

Identification

This document, IBM Contract Data Requirements List (CDRL) Sequence Number 1490A, is the delivery for STARS Delivery Order Task R20 (Process/Environment Integration).

Purpose

This document is an update to the document, IBM Contract Data Requirements List (CDRL) 1490, Application Blueprint Definition for C3. It describes the application blueprint concept and a first pass of the domain analysis phase of an application blueprint for the selected C3 application, Air Traffic Control Systems. The updates to CDRL 1490 include additions to the domain analysis and generic architecture discussions, and incorporation of the peer review comments.

Acknowledgements

The work for CDRL 1490A was performed by Joanne Piotrowski under the scope of task IR20. Technical reviewers for CDRL 1490A were Jim Moore, Mary Catherine Ward, and Gary Turner. The original document, CDRL 1490, was the result of work done by Joanne Piotrowski under the scope of task IR20, with assistance from the technical reviewers Dick Drake, Jim Moore, and William Ett. The generic Information Object Model (IOM) of an Air Traffic Control System, used as a basis for this work, was a result of IQM15 task work using the Foxboro Information Object Model Methodology. The IQM15 team included William Ett (chief programmer), Marjorie Conley, Barbara Hoeper, Kimberly Lesho, Robert Simonoff, and Joanne Piotrowski (participating as an IR20 team member for the first part of the IQM15 Phase II task).

Abstract

Application Blueprints serve as a framework for designing new systems in an application domain, leading to reuse of design information and greater reuse of code. This document discusses the definition for an application blueprint. The process used to create a blueprint is outlined, and the benefits and drawbacks of an application blueprint are discussed. The Appendix of this document presents the first pass of the domain analysis and generic specification of an Application Blueprint for an Air Traffic Control System. This paper should be used as the groundwork for future STARS work in the reuse of analysis and design information.

Introduction

Most of the components reused today are small and general enough to be adapted for use in many different systems. Although some productivity is gained by the reuse of small components, reuse of non-domain specific components frequently requires a large amount of new code to glue the components together and to tailor the components to the domain. Reuse of small pieces of code is not going to substantially improve the productivity of systems development. A greater improvement in the productivity and quality in systems development can be achieved by reusing design information.

Studies of the techniques and work products of expert designers show that they typically do not start their designs from scratch. Instead they base their new designs on previous knowledge, using the same patterns repeatedly in their designs. "They are bringing a large amount of pre-structured information (i.e. partially-specified architectures) to bear on the problem" (Bigg87a).

Application blueprints are a means for promoting reuse of analysis and design information within a domain, resulting in a significant improvement in the productivity and quality of system development. Reusable architectures will reduce the design time for the total system, allowing developers to concentrate on designing the unique features of each system. Also, because the same general framework has been reused, larger, integrated collections of components can be reused in these systems, reducing the total development time for the system.

Application blueprints serve as a framework for designing new systems in an application domain. A standard set of interfaces is provided by the application blueprint for the domain. These interfaces support greater reuse of analysis information, design, and code.

What is an Application Blueprint?

Before the concept of an application blueprint can be introduced, one needs to understand the concept of a domain and the process of performing a domain analysis.

Domain

A domain is an "encapsulation of a problem area" (Neig84). It is a set of current and future systems/subsystems marked by a set of common capabilities and data. (SEI)

(Quan88) has identified two types of domains - a problem domain and an architectural domain. The problem domain contains all applications that are part of the domain because they share common functions. An architectural domain contains all domain applications that are in the problem domain and also can share the same high level generic architecture and set of reusable components. The problem domain may contain several architectural domains.

A problem domain may be composed of multiple domains that are less application specific. "While some domains sound very problem specific such as an Air Defense System domain...these domains are primarily built out of general and much more reusable domains (ex. database domains) which are tailored in the refinement process to the specific problem." (Neig84)

Domain Analysis

Domain analysis is a process by which "common characteristics from similar systems are generalized, objects and operations common to all systems within the same domain are identified, and a model is defined to describe their relationships." (Prie87) "Domain analysis can be seen as a process where information used in developing software systems is identified, captured, structured, and organized for further reuse." (Prie90) "Components that result from domain analysis are better suited for reusability because they capture the essential functionality required in that domain." (Prie87) The domain analysis process involves extensive consultations with experts in the domain and analysis of existing domain applications.

There are many research efforts going on to define a methodology for performing the domain analysis (e.g. the work of the Software Reuse Group at the SEI, Pittsburgh, PA). These efforts are based on applying known techniques from the fields of artificial intelligence and systems analysis to the area of domain analysis. Knowledge acquisition and knowledge representation techniques of artificial intelligence can be used for obtaining the domain knowledge. System analysis techniques, typically used for analyzing specific system requirements, can be broadened for use in analyzing requirements for multiple systems in a domain (Prie90).

Application Blueprints

Application blueprints are a means of capturing analysis, design, and implementation information for a specific domain in a form which facilitates reuse in that domain.

A blueprint for a house is used to present the plan for building a house. A builder can create many different houses from the same basic model presented in the blueprint by customizing the house. The "interfaces" for the builder's components are specified in the blueprint. Specific options can be selected such as adding extra rooms or selecting different styles of windows and doors that fit the blueprint specifications.

Similarly, an application blueprint can be used to present the base design for specific applications in a domain. The generic model presented in the blueprint can be customized to create a specific application by adding new functions and selecting different components that meet the interface requirements of the application blueprint and fit the needs of the specific application.

The application blueprint provides a way of standardizing the software interfaces for an application domain, similar to the way hardware component interfaces have been standardized. This promotes greater reuse within the domain by increasing the probability of finding acceptable parts that can be reused as black boxes. By formalizing the standardization, the likelihood of finding a part will be increased and an inter-organizational sharing of parts will be encouraged.

The application blueprint consists of the following parts:

- *domain analysis information* - notes from the domain analysis process
- *generic functional specification* - requirements for the high-level design
- *generic high level design* - the generic architecture, providing the base for the component structure, i.e. the framework for integrating the set of domain specific reusable components
- *set of highly integrated, domain specific reusable components*
- *generic architecture prototype* - an implementation of the top level structure of the generic system that will be used as the initial prototype for specific applications in the domain
- *design and implementation information* - Notes on design rationale and tradeoffs

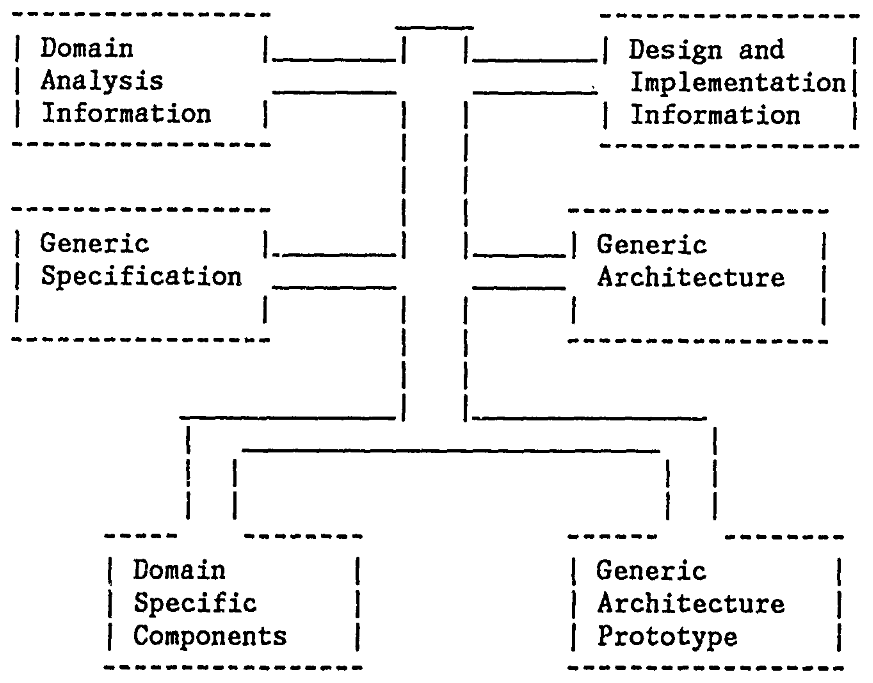
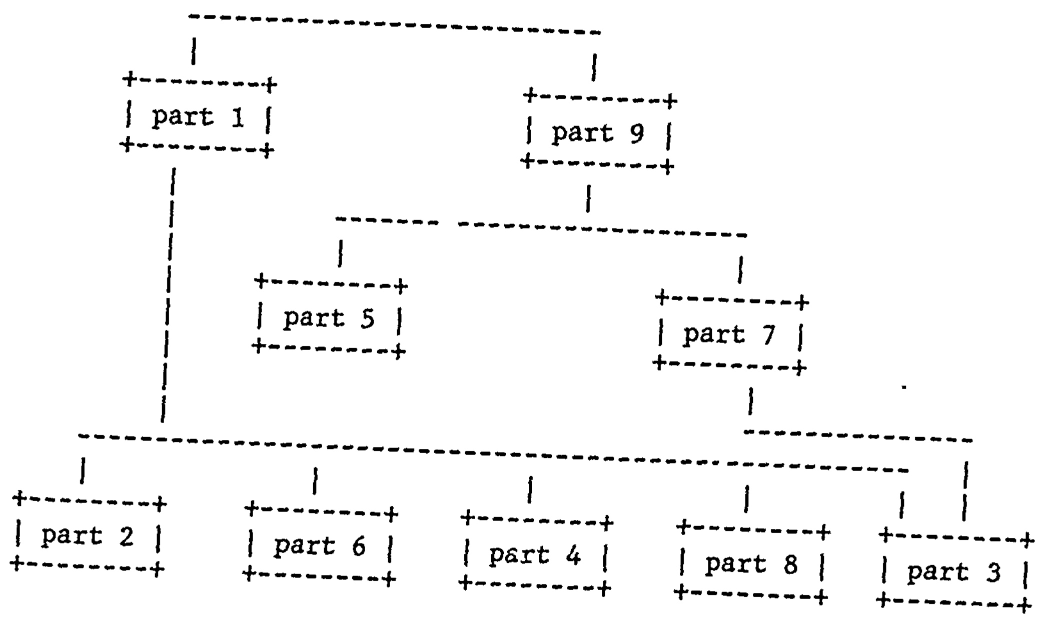

Domain Analysis Information
Design and Implementation Information
Generic Specification
Generic Architecture
Domain Specific Components
Generic Architecture Prototype

Figure 1. Application Blueprint

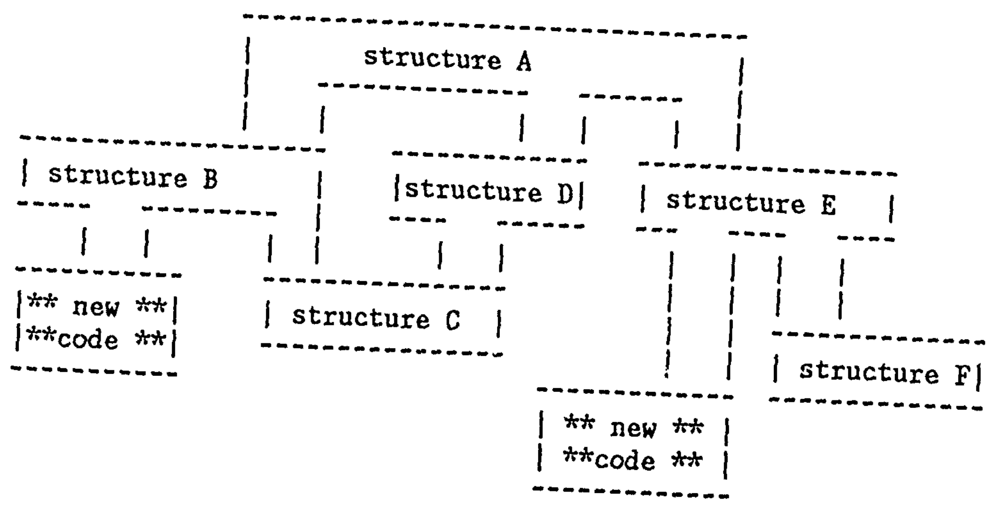
Figure 1 emphasizes the strong mapping between the parts of an application blueprint. Two-way traceability between any two of the pieces of the blueprint is important so that the entire blueprint can be developed as a single major work product using prototyping. The parts within an application blueprint are all developed concurrently by a multi-disciplinary team. Each team member views the system from his own discipline. These views must be consistently represented in each part of the application blueprint. Changes must also be replicated through each part of the blueprint, so each piece of the blueprint is consistent with the other pieces.

(Boll89) distinguishes two types of reuse - compositional and adaptive - that are important to the reuse promoted by application blueprints. Compositional reuse occurs when individual components are used to develop the application, typically in a bottom-up fashion. New application specific code is written to interconnect the individual parts retrieved from the component library (see Figure 2). Adaptive reuse encourages a top-down approach to development, with higher level structures obtained from the reuse library. New application specific code is added at lower levels of the system (see Figure 3). This form of reuse preserves architectural stability by leaving the overall framework of the system unchanged.



(adapted from Boll89)

Figure 2. Compositional Reuse



(adapted from Boll89)

Figure 3. Adaptive Reuse

Application blueprints promote a hybrid of compositional and adaptive reuse. Each structure and new piece of code of the adaptive reuse system can be created using compositional reuse techniques. For example, **Structure C** in the adaptive reuse example in Figure 3 could be composed of the **connected parts** in the compositional reuse example Figure 2.

Advantages and Drawbacks of Application Blueprints

Advantages

The greatest benefit of Application Blueprints is that it leads to a high level of reuse. By orienting the components to an application domain, the components can be larger, more complex, and highly integrated, leading to greater reuse within the domain. (Quan88) The payoff involved in reusing a component increases more than linearly as a component grows in size (Bigg87a). Studies have shown that "technologies that are very general, in that they can be applied to a broad range of application domains, have a much lower payoff than systems that are narrowly focused on one or two application domains" (Bigg87a). The functions and interfaces of the domain components are well documented for specific applications, making it easier to use the components in the domain. Domain specific components are more efficient than components that try to be useful to many domains.

The effort involved in developing a specific application from an application blueprint is much less than developing the system from scratch. Since the framework of the application is reused, there is a minimal design effort involved in tailoring for the specific application. Coding and testing efforts are reduced because large components of the system have already been developed and tested. Testing efforts can also be reduced by developing a generic test harness for applications in the domain.

The application blueprint promotes greater reusability within domains by providing a standard software interface to the functions of applications in the domain. If this standardization can be formalized by the industry, the amount of reusable analysis, design, and code will be greatly increased and there will be a greater potential for finding acceptable parts for the applications. This will bring software reuse closer to the reusability that is achieved with hardware components.

There is a higher degree of quality in the specific applications developed from application blueprints. The design of the specific application is derived from the generic architecture of the application blueprint, which was created by experienced designers in the domain. This design has to take into account all of the lessons learned of the experienced design team to produce a design of higher quality. The components in the application blueprint have been coded for reuse and should have few defects. As the components are used in specific applications, the defects in the components are reduced.

The application blueprint provides a "consistency in the implementation and behavior of the applications that share the architecture." (Quan88) Developers working in the same domain can rapidly understand the complete design of specific applications. The blueprint encourages use of a common terminology throughout the domain. Maintenance activities are easier because maintainers familiar with the application blueprint can easily maintain the specific applications in the domain.

The application blueprint provides good support for early prototyping activities. A working generic application that implements the major components of the system can be used as an early prototype for the specific applications. Specific application developers can add, delete, and modify functions of the generic prototype to develop the specific application prototypes.

Drawbacks

The major drawback of the application blueprint is the cost of developing the generic system. The entire domain must be understood so that common functions can be determined and future changes anticipated. The lifespan and the potential uses of the application blueprint must be studied to justify the cost of developing an application blueprint. Development of a generic architecture is considerably more expensive than developing a specific application because of the additional analysis required to represent the requirements of many applications and the increased effort to develop a flexible design that can be adapted for specific applications and future requirements.

Using an application blueprint can make it harder to adapt to changes in technology because each change affects many specific applications. Technology changes need to be planned for early in the development of the application blueprint.

The domain-specific components of the application blueprint are less reusable outside of the domain. Since the domain-specific components perform domain functions and are highly integrated within the generic architecture, they are harder and less cost effective to use outside of the domain. "Modules become less and less reusable the more specific they become, because it becomes more and more difficult to find an exact (or even close) match of detailed specifics" (Bigg87a).

Organization of Software Development Groups

Current software development roles need to be altered to support the development and use of an application blueprint. There should be two main development roles within a software organization, the domain engineers and the application engineers.

The domain engineers are experienced in the domain analysis process and are considered experts in the domain. They are skilled in the techniques of abstraction and domain analysis. Domain engineer responsibilities include identifying the domain, performing the domain analysis, and creating the application blueprint.

The application engineers are responsible for developing an application in the domain. They use the software assets developed by the domain engineers and obtain domain knowledge from the domain engineers. The application engineers need to be skilled in prototyping techniques. The application engineers provide feedback to the domain engineers to improve the application blueprint for use by future domain applications.

```
===========
| DOMAIN  |
| INFO    |                                             ----> SPECIFIC
===========                                             |     APPLICATION
     |                -----------------                 |         X
     |                |               |                 |
     V                |  APPLICATION  |                 |
 DOMAIN    --------->|   BLUEPRINT   |----------> APPLICATION ------
 ENGINEERS            |               |             ENGINEERS        |
    /\                -----------------                 |            |
     |                                                  |            |
     |                                                  |            V
     ---------------------------------------------------         SPECIFIC
        feedback of: blueprint change requests,                  APPLICATION
                     new domain knowledge,                           Y
                     new domain components
```

Figure 4. Participants in Software Development Efforts

Two distinct life-cycle processes are required to support the development of application blueprints and their use in developing specific application. Application development efforts typically do not have enough time, money, or resources to develop reuse assets for payoff in later application development efforts. Development of an application blueprint is a complex and costly effort requiring additional investment. Companies need to realize that the investments in application blueprint efforts will payoff many times in future efforts (Drak90).

Creating an Application Blueprint

It is recommended that domain engineers utilize a multi-disciplinary team to create the application blueprint so that multiple views are represented in the work product. Because the blueprint will be the foundation of the specific applications in the domain, the team should be highly experienced in both software development and the domain, so that their expertise can be captured for use in the specific applications of the domain.

The process for creating an application blueprint is shown in Figure 5 on page 11.

```
  -------     ----------------------------------------
  |           |                                      |
  |           |         Identify the Domain          |
  |           |                                      |
  |           ----------------------------------------
PHASE I -                         |
DOMAIN                            V
MODELLING    -----------------------------------------
  |           |                                      |
  |           |       Perform Domain Analysis        |
  |           |                                      |
  -------     ----------------------------------------
                                  |
                                  V
  -------     ----------------------------------------
  |           |                                      |
  |           |   Produce a Generic Specification    |
  |           |                                      |
PHASE II -    ----------------------------------------
DOMAIN                            |
ARCHITECTURE                      V
  |           ----------------------------------------
  |           |           Build a Generic            |
  |           |          High-Level Design           |
  |           |                                      |
  -------     ----------------------------------------
                                  |
                                  V
  -------     ----------------------------------------
PHASE III    |     Prototype a Generic System and    |
SOFTWARE     |     Develop the Set of Reusable       |
ASSET        |     Components for the Domain         |
DEVELOPMENT  -----------------------------------------
  -------
```

Figure 5. Generating an Application Blueprint: Note that feedback loops are not shown on the diagram, but occur between each of the steps.

There are three phases to developing an application blueprint:

- Domain Modelling, where the domain is identified and the domain analysis is performed
- Domain Architecture, where the generic specification and design is produced
- Software Asset Development, where the prototype generic system and domain specific components are developed

The steps required for completing each phase of the application blueprint are described below.

- PHASE I: Domain Modelling

  1. Identify the domain

     The domain should be relatively stable and contain a large number of applications. The boundaries of the domain need to be identified and interfaces to the external entities defined. In addition to defining and scoping the domain, an approach to the domain analysis and the sources of knowledge and information about the domain are identified. This step also includes performing a cost justification for developing the application blueprint. There should be enough applications in the domain to justify the upfront cost of developing the blueprint. But, it may be beneficial to create an application blueprint for a small domain, even a domain with a single application, in anticipation of future needs for a generic architecture.

     Several guidelines should be used when defining the domain for an application blueprint:

     - The applications in the domain should perform common functions so that they can share a common architecture.
     - The domain should be defined so that there are sufficient applications within the domain that share common functions to justify the development cost of a generic architecture. (Quan88)
     - The domain should be sufficiently large and stable. "A large static domain means that the components will be useful for longer periods of time, justifying the cost of creating the components. The worst kind of domain for reusability is one where the underlying technology is rapidly changing". (Bigg87a) A rapidly changing domain should not be eliminated until it is determined whether the high level generic architecture for the domain would remain stable, even though the lower level components change over time.
     - The hardware and software environments for the domain applications should be similar to increase the chances for reusability between domain applications.

  2. Perform Domain Analysis

     In the domain analysis stage, the common characteristics of the application in the domain are identified through consultation with experts in the domain and examination of existing applications in the domain. (Prie87) This is the most critical part of the application blueprint development.

     It is important that enough effort be used for the domain analysis since it is the foundation upon which the application blueprint is built - the identification of common requirements for applications in the domain. From experiences in performing ten domain analyses, Neighbors has stated that it typically "takes an expert in a particular problem area four months to complete a first attempt at the domain." (Neig84) "Careful consideration over a long period of time is required to produce a model of objects and operations in a problem area which is flexible enough to last through the ten to fifteen year lifespan of a large system." (Neig84) The lifespan of defense projects is typically longer than this, twenty to thirty years. The objects and operations of the domain analysis must be selected so that they are flexible enough to last the lifespan of the system.

Two types of domain analysis should be performed during the domain analysis stage: horizontal domain analysis and vertical domain analysis. "Horizontal domain analysis identifies components that can be reused across a broad range of application areas, such as data structures and user interface mechanisms. Vertical domain analysis focuses on isolating portions of a particular application that can be reused in different versions of similar applications within the scope of the same problem domain." (Trac87a)

- PHASE II: Domain Architecture

  1. Produce the Generic Specification

     Use the information collected in the domain analysis to produce the generic specification. The specification describes the objects and operations common to applications in the domain. The specification must be written in the language of the domain (i.e. domain terminology) so that it can be reviewed and understood by the experts in the domain.

  2. Build the Generic High-Level Design

     The generic high-level design is the framework for designing specific applications. It provides a standard set of interfaces that can be easily adapted to support the requirements of the specific applications in the domain.

     Object oriented design techniques seem to be the best way to enforce this mapping between the parts of the application blueprint. The domain analysis would identify objects and classes of objects, as well as operations on the objects. By using object oriented techniques, the generic architecture for the domain will be more likely to remain stable as the system changes. The scope of the changes can be isolated to individual objects of the domain that are loosely coupled to the other objects in the system. Because of the loose coupling of objects, it is likely that the effect of generic architecture changes, for example, adding a new requirement to the generic system or adapting the generic architecture to the specific application, would only affect a small, known portion of the existing system.

- PHASE III: Software Asset Development

  1. Prototype the Generic System and Develop the Set of Reusable Components

     The implementation of the generic system consists of a top level structure for the domain and a set of reusable components specific to the domain. Because these components are domain specific, they should be relatively large. The domain components must be thoroughly documented to describe the usage of the components within the applications. The implementation of the generic framework is the initial prototype of the specific application development.

Note that it is not necessary to complete each step of the process before proceeding to the next step or next phase of the process. A better method would be to go through the steps several times, using rapid prototyping. This gives the domain experts early results for review.

Also note that sometimes it will be necessary to repeat previous steps to account for new knowledge about the domain. For example, when performing the domain analysis, the identification of the domain will need to be reconsidered because domain boundaries could change as more is learned about the domain. As the generic specification is produced, more domain information may need to be collected through domain analysis.

Development of a generic application blueprint is an ongoing process. As applications are generated from the application blueprint, the application blueprint should be updated to incorporate the new domain knowledge. In particular, the collection of domain specific reusable components needs to be updated to include the newly developed components and increase the library of options for future applications.

Using an Application Blueprint

Figure 6 on page 14 shows how an application blueprint would be used by application engineers to develop a specific application in the domain.

Given that a generic architecture and a set of domain specific components exist for the domain, a developer of a specific application in the domain would use the i· :plementation of the generic architecture as the initial prototype of the system. Customer feedback would be obtained and the prototype would be modified to include the application specific requirements. Specific application changes could be newly coded or obtained from the set of domain specific and general components in the repository. Any newly developed code with a potential for reuse should be submitted to the repository. Also, any new domain knowledge should be shared with the developers of the application blueprint for incorporation into the blueprint. When the customer has accepted the prototype, the prototyping phase of development will be completed and the developers will productize the system.

```
     ------------------------------------
     |         Develop a generic        |
     |         architecture             |
     |         for an application       |
     |         domain                   |
     ------------------------------------
set of domain |      | high     | high level
specific      |      | level    | generic
reusable      |      | generic  | implementation
components    |      | design   |
              V      V          V
         =============================
         (       REPOSITORY          )
         =============================
           /\                | high level
           |                 | generic
           |                 | implementation
           |                 |
           |                 V
domain     |          --------------------
specific   |          | Demonstrate      |
reusable   |          | Prototype        |<---------------
components |          | to the           |               |
    +      |          | Customer         |               |
general    |          --------------------               |
reusable   |             | Customer                      |
components |             | Decision                      |
           |             |            ---------------    |
           |             V            | Productize  |    |
           |                   yes    |    the      |    |
           |          accept?  ------>| Specific    |    |
           |                          | Application |    |
           |                          ---------------    |
           |             |                   |           |
           |             |                   |           |
           |             |no                 |           |
           |             V                   V           |
           |      ----------------      SPECIFIC         |
           |      | Identify     |      APPLICATION      |
           |      | Required     |                       |
           |      | Changes to   |                       |
           |      | Prototype    |                       |
           |      ----------------                       |
           |             |                               |
           |             V                               |
           |      ----------------                       |
           |      | Implement    |                       |
           -----> | New Version  |------------------------
                  | of Prototype |
                  ----------------
```

Figure 6. Using an Application Blueprint

Use of the IOM Methodology for Domain Analysis

Many methodologies can be used to perform the domain analysis. We developed the Air Traffic Control Application Blueprint using the Foxboro Information Object Model Methodology for the domain analysis to evaluate its usage in the development of the application blueprint.

Note: The Overview of the Information Object Model was based on talks given by Dr. Gerald White of Foxboro Corporation.

Overview of Information Object Model Methodology

The Information Object Model (IOM) Methodology was developed to solve the problems with using traditional structured analysis. It is modeled after the Ward/Mellor structured analysis techniques. The major features of the IOM methodology are the interview techniques used to perform the domain analysis and the layered presentation of information. The layering of information forces the analyst to push the details of the system to lower levels, concentrating on the specifications of the system without implementation concerns; the "whats" of the system are considered before the "hows". Functional deliverables are a requirement of the methodology. Graphical techniques and the use of multiple views of information are important tools for recording and presenting the analysis information. Expert system knowledge acquisition techniques are used to understand the functions in the domain.

The initial work products of the Information Object Model Methodology are:

- Mission Statement

  The Mission Statement is a clear concise (one page) statement of what the system does and performance requirements. It is used to focus the team on the task and define the boundaries of the task. Customers can also use it to reach agreement on the task definition.

- Overview

  The Overview is a ten page document that presents an overview of the system in domain terminology. It is used to establish the system boundaries and constraints on the model. It also describes the major functions of the system.

- Information Object Model

  The Information Object Model is a layered structured analysis of the system. It presents multiple views of the information, combining graphics, a data dictionary, and user generated text to functionally describe the requirements of the system. It is the functional specification of the system. The Excelerator tool was used to record the IOM information.

The domain information for the IOM is obtained through interviews of domain experts. A development team with no previous experience in the domain should be used to reduce implementation biases in the early phases of the IOM. Through multiple interviews of the domain experts and presentations of multiple views of the acquired domain information to the domain experts for review, the developers quickly become knowledgeable in the domain.

Applicability of IOM Methodology to Application Blueprints

One of the benefits of the methodology is that the blueprint developer who is new to the domain can quickly "come up to speed" on the domain using the IOM Methodology. Interviewing experts instead of reading volumes of information are important features of the methodology. The knowledge of the experts can be used to navigate the developer quickly through the domain documentation.

A second benefit of the methodology is its use of data flow diagrams to provide a layered presentation of the domain information in the terminology of the domain. These diagrams and accompanying documentation can by easily understood by the experts of the domain so that the developer can get validity checks from the experts.

The initial three IOM Methodology work products correspond to the first steps involved in creating an application blueprint. The Mission Statement for a generic system defines the domain and identify the domain boundaries. This is one part of the domain identification step, the other part being the cost justification for the development of the application blueprint for the domain.

The Overview would be a work product of the early stages of the domain analysis task. It further defines the domain and identifies the major functions of the domain. The Generic Specification step is represented in the IOM Methodology as the Information Object Model. It is the functional specification for the generic system. All of the common functions of the domain are represented in the IOM.

For a complete discussion of the IOM Methodology see IBM CDRL 1200A, *Information Object Modeling Methodology*.

Lessons Learned

The generic application blueprint should never be considered completed. As specific applications are created from the application blueprint, the generic model should be revisited to add new insights. Also, as new components are developed for specific applications, the components should be added to the application blueprint's collection of domain-specific components.

Many methodologies could be used to develop the blueprint. For the example Air Traffic Control application blueprint, we tried the IOM Methodology to perform the domain analysis and create a generic specification. It promotes reusability and gives a clear, layered approach to the domain analysis and systems development. It would be beneficial to experiment with other methodologies.

A highly experienced team needs to develop the application blueprint. The design and domain expertise of the blueprint developers is critical to the quality of the application blueprint. The developers need to have enough experience in the domain to anticipate technology changes, so the application blueprint can be useful for long periods of time.

Many domains contain smaller, more reusable domains. Domains that appear within many more specific domains need to be identified. These domains can be considered subsystems within the specific application. Application blueprints for these domains would provide a standard interface to the subsystems and maximize reuse. For example, the Air Traffic Control System needs to maintain many databases. If a database application blueprint existed with a clearly defined standard interfaces, then it could be tailored to the ATC needs and the Air Traffic Control System application blueprint could interface to the subsystem.

The development of an application blueprint can easily be tied to the Software-First Life Cycle (SFLC) (IBM1240) phases of development. The domain analysis, generic specification, and high level generic architecture development are performed during the Preliminary System Analysis phase of the SFLC. The prototypes for the generic architecture would be developed during the System Architecture phase of the life cycle. For a further discussion of the application blueprint in the SFLC see *Software-First Life Cycle Final Definition* (IBM1240).

Recommendations for Future Application Blueprint Work

The following is a list of recommended areas for further research of the application blueprint concept.

- A machine processable representation for the blueprint design information needs to be developed.

  A major hindrance to the reuse of design information is the lack of a machine processable representation of the design information. One of the desired features of a design representation is that it have "the ability to create partial specifications of design information, specifications which can be incrementally extended" (Bigg87a). Existing specification languages require that all details of the system be specified. The idea of a partial specification is particularly important to the application blueprint because the generic architecture provides the overall framework of the system which is refined as specific application requirements are added to the design. The representation must allow for partial specification of the details of the system so that the design will be generic and reusable. "If we insist on specifying the details too precisely, we over commit to the details of the resulting component, thereby reducing its reuse potential. If we leave too many of the details completely unconstrained, we have significantly fewer hooks for automation and we reduce the payoff of reusing the component because so much manual labor is involved. Without a representation that allows a mixture of precision and fuzziness, we lose much of the advantage of reuse" (Bigg87a).

  The design representation language must also have "the ability to allow flexible couplings between instances of designs and the various interpretations those instances can have" (Bigg87a). "We want to represent the essence of a design component (factor) rather than its details, permitting us to apply concepts taken from one domain to structures within an entirely different context." (Bigg87a) The domain language must have referential flexibility, with domain terms referred to by semantic terms rather than syntactic names, so that the design can be easily used within multiple domains.

  Some research is being performed by Dr. Mary Shaw (SEI, Pittsburgh,PA) to investigate ways to describe and specify architectures.

- The retrieval of domain specific reusable components from the repository should be studied.

  Users of a parts repository typically search the repository to retrieve a single part that meets their needs. In contrast, application blueprint users will search the repository and retrieve the collection of domain specific components. A single retrieval request should be used to retrieve whole subsystems of the domain. Domain specific retrieval mechanisms should guide the application developer through the development process. The majority of the components in the domain collection will be reused in the specific application. Organization of the repository by domain should also be studied.

- As the implementation model and the project phase of the Foxboro IOM methodology is defined, it needs to be examined for its applicability toward the creation of application blueprints.

- The reuse of application blueprints within more specific application blueprints needs to be investigated.

For example, the reuse of a database application blueprint within the Air Traffic Control application blueprint.

- After completion of an application blueprint for a domain, an experiment should be conducted to develop specific applications in the domain to test the concept and study the amount of reuse that is achieved from use of the application blueprint and the gains in productivity.

Further work is needed to complete the Air Traffic Control IOM so that the high level architecture can be developed and the application blueprint completed. This includes the further decomposition of the generic functions to the primitive level.

Glossary

adaptive reuse: A form of reuse where higher level structures with clearly defined interfaces are obtained from a reuse library. New code is written for the lower level functions.

application blueprint: A means of capturing analysis, design, and implementation information for a specific domain in a form which facilitates reuse in that domain. The blueprint includes a generic functional specification, a generic high level design, and a set of domain specific reusable components.

architectural domain: A classification for a domain that contains a set of applicaitons that share common functions and can use a common architecture.

compositional reuse: A form of reuse where individual components are used from a library of components. New code is written to "glue" the components together.

domain: Encapsulation of a problem area.

domain analysis: Common characteristics from similar systems are generalized, objects and operations common to all systems within the same domain are identified, and a model is defined to describe their relationships. (PRIE87)

generic architecture: A high-level design that define standard interfaces for applications within a domain.

horizontal domain analysis: "Identifies components that can be reused across a broad range of application areas, such as data structures and user interface mechanisms." (Trac87a)

Information Object Model (IOM) Methodology: A methodology based on Ward/Mellor structured analysis techniques that promotes knowledge acquisition techniques and a layered presentation of information.

problem domain: A classification for a domain that contains a set of applications that share common functions.

vertical domain analysis: "Focuses on isolating portions of a particular application that can be reused in different versions of similar applications within the scope of the same problem domain." (Trac87a)

Acronyms

Acronym	*Meaning*
ATC	Air Traffic Control
CDRL	Contract Data Requirements List
IBM	International Business Machines
IOM	Information Object Model
SFLC	Software-First Life Cycle
SOW	Statement of Work
STARS	Software Technology for Adaptable, Reliable Systems

Bibliography

(Bass87)	Bassett, P. "Frame-Based Software Engineering", *IEEE Software*, July 1987.
(Bigg87a)	Biggerstaff, Ted and Richter, Charles "Reusabiity Framework, Assessment, and Directions", *Proceedings of the Twentieth Annual Hawaii International Conference on System Sciences*, 1987.
(Bigg87b)	Biggerstaff, Ted and Richter, Charles "Reusability Framework, Assessment, and Directions", *IEEE Software*, March 1987.
(Boll89)	Bollinger, Terry B. "Making Reuse Cost Effective", Foils from *Reuse and the S/W Revolution Symposium* February 1989.
(Booc87)	Booch, G., *Software Engineering with Ada*, The Benjamin/Cummings Publishing Company, Inc., 1987.
(Drak90)	Drake, Richard J., *The Domain Development Process*, IBM Federal Sector Division, May 29, 1990, unpublished paper.
(Horo84)	Horowitz, E. and Munson, J. "An Expansive View of Reusable Software", *IEEE Transactions on Software Engineering*, Vol Se-10 #5, Sept 1984.
(IBM1200A)	IBM Systems Integration Division, *Information Object Modeling Methodology*,CDRL Sequence No. 1200A, June 1990.
(IBM1240)	IBM Systems Integration Division, *Software-First Life Cycle Final Definition*,CDRL Sequence No. 1240, January 1990.
(Kais87a)	Kaiser, G.E. and Garlan, D. "Composing Software Systems from Reusable Building Blocks", *Proceedings of the Twentieth Annual Hawaii International Conference on System Sciences*,1987.
(Kais87b)	Kaiser, G.E. and Garlan, D. "Melding Software Systems from Reusable Building Blocks", *IEEE Software*, July 1987.
(Lane84)	Lanergan,R. and Grasso, C. "Software Engineering with Reusable Designs and Code", *IEEE Transactions on Software Engineering*, Vol SE-10, #5, p498-501, Sept 1984.
(Lenz87)	Lenz, M, Schmid, H.A., Wolf, P. "Software Reuse through Building Blocks", *IEEE Software*, July 1987.

(Mats84) Matsumoto, Y. "Some Experiences in Promoting Reusable Software Presentation in
Higher Abstract Levels", *IEEE Transactions on Software Engineering*, Col. SE-10, No.5, Sept. 1984.

(Neig84) Neighbors, James "The Draco Approach to Constructing Software from Reusable Components", *IEEE Transactions on Software Engineering*, Vol SE-10, #5, p564-574, Sept 1984.

(Prie87) Prieto-Diaz, Ruben "Domain Analysis for Reusability", *Eleventh Annual International Computer Software and Applications
Conference, COMPSAC 87*, Oct. 7-9, 1987, Tokyo, Japan.

(Prie90) Prieto-Diaz, Ruben "Domain Analysis: An Introduction", Draft Paper, March 7, 1990, unpublished paper.

(Quan88) Quanrud, Richard B., "Generic Architecture Study", SOFTECH, Inc., Prepared for U.S. Army CECOM, Ft. Monmouth, NJ, Report 3451-4-14/2, January 22, 1988.

(Trac87a) Tracz, Will "RECIPE: A Reusable Software Paradigm", *Proceedings of the Twentieth Annual Hawaii International Conference on System Sciences*, 1987.

(Trac88) Tracz, Will "Software Reuse Myths", *Software Engineering Notes*, vol 13, no 1, Jan 1988, ACM SIGSOFT.

Appendix A. Application Blueprint for Air Traffic Control Systems

The first pass of the generic Air Traffic Control (ATC) Mission Statement and Information Object Model is included in this appendix as the generic specification of the application blueprint for Air Traffic Control Systems. It was the result of the Phase 2 work of the STARS IQM15 team work. It consists of Excelerator graphs and reports.

Domain knowledge was obtained through numerous ATC documents and interviews of three domain experts, who were former controllers. One of the experts was interviewed at the beginning of the process to obtain an overview of the ATC systems. This information helped in writing the mission statement for the generic ATC systems. The second expert was asked to review the mission statement and to provide information about the major functions of the generic ATC system. This information was used to prepare the Information Object Model. The third expert provided additional information about the domain and critiqued the first pass of the generic ATC Information Object Model. Since the IOM development team had little ATC domain knowledge, the feedback from the domain experts was important to ensure the validity of the generic ATC IOM.

To complete the ATC application blueprint the generic IOM needs to be completed. Each major function of the IOM needs to be decomposed to the primitive level. After the IOM is completed, the generic architecture can be designed.

For further information about the IOM task and the generic Air Traffic Control System, see STARS IQM15 CDRL 1200, *STARS Structured Analysis for Selected Application.* STARS IQM15 CDRL 1200A, *Information Object Modeling Methodology* discusses the IOM methodology.

Mission Statement: Air Traffic Control

The proposed mission of an air traffic control (ATC) system is to control and monitor an area of defined air space and to control terminal ground operations. This system provides for the safe and timely departure and arrival of controlled flights.

An ATC system will provide monitoring of aircraft positions in relation to other aircraft traffic, terrain, and weather conditions within a defined air space. An ATC system will provide an air space situation assessment capability to identify a situation for the control function to manage.

Control for an ATC system will provide:

- sequencing and separation of aircraft,
- navigation instructions to avoid identified situations (e.g. conflict alerts, hazardous weather, terrain obstacles),
- for the tracking of controlled aircraft against filed flight plans,
- navigation instructions to aircraft as requested.

Terminal ground operations for an ATC system includes control of ground travel and issuance of takeoff/landing clearance.

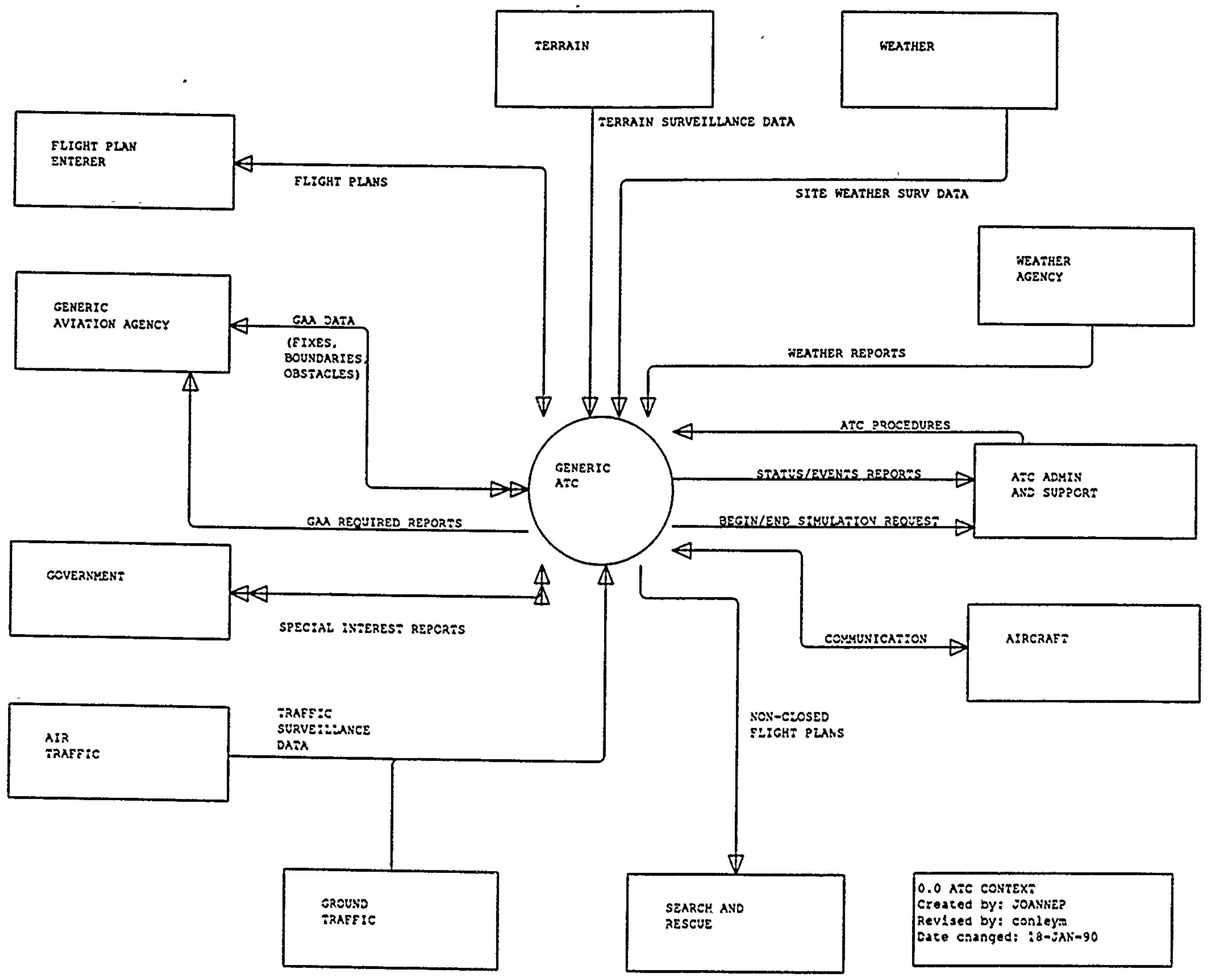
An ATC system will also:

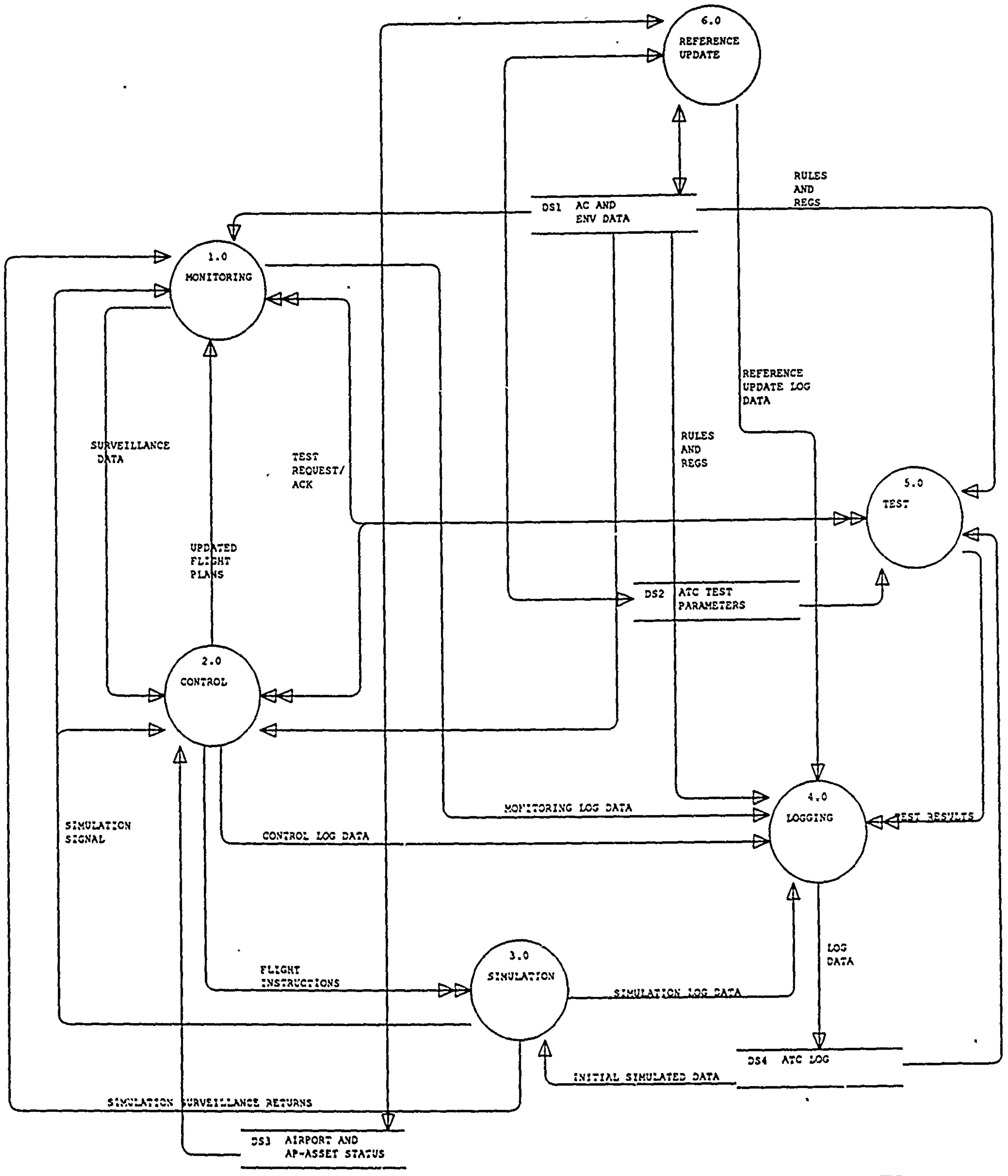
- accept, process, allow in-flight modification and closeout of flight plans,
- provide communication between controller and aircraft,
- provide weather information and re-route controlled traffic accordingly,
- provide traffic re-routing due to exceptional conditions (e.g. aircraft emergency, airport closing, etc.).

A typical scenario for operating an aircraft under the guidance of an ATC system includes the following:

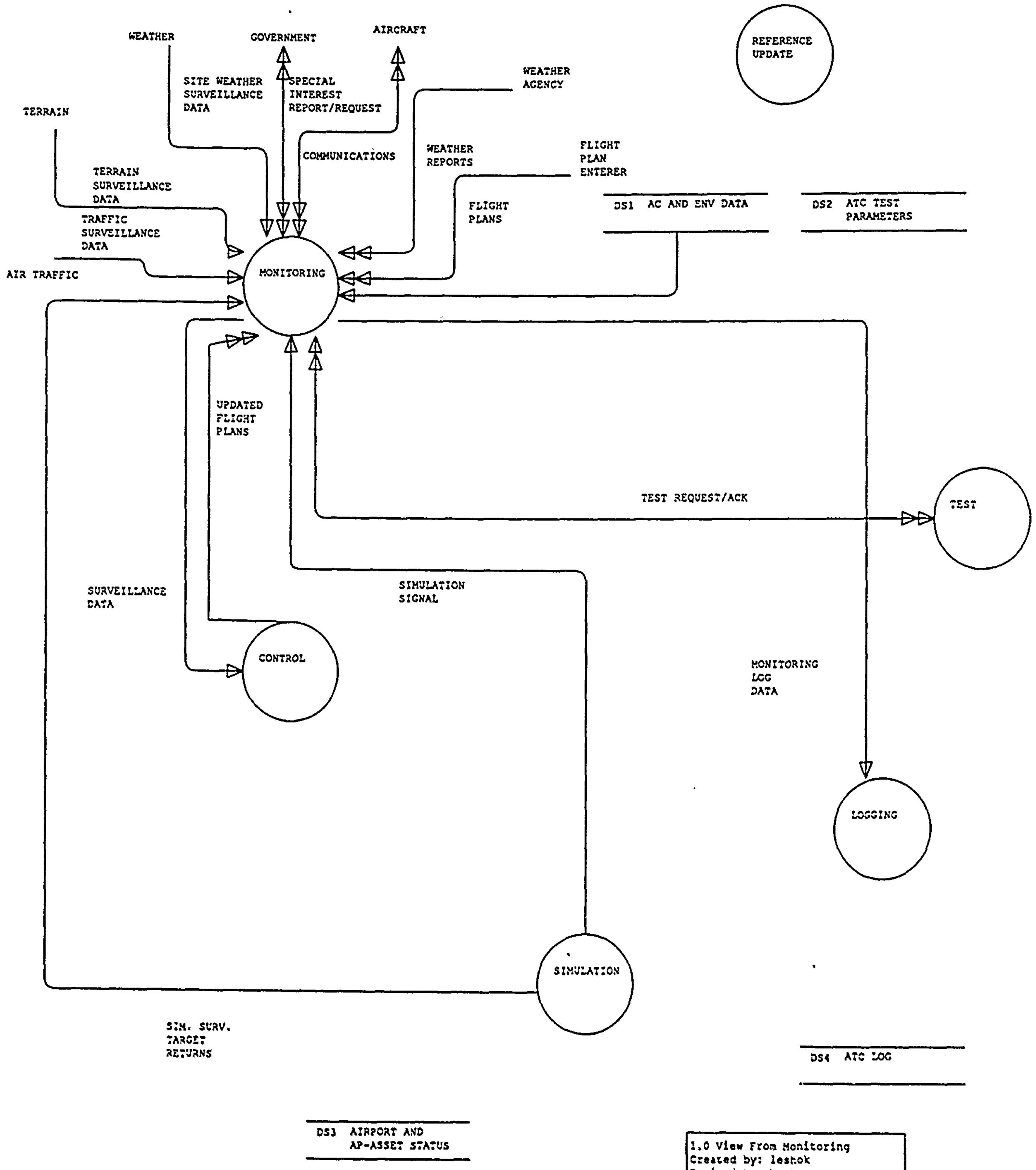
- Pre-Flight
    - Flight plan entry
- Departure
    - Push back clearance
    - Taxi clearance
    - Take-off clearance
- In-Flight
    - Spacing, monitoring, and tracking of aircraft
- Approach
    - Sequencing and spacing of aircraft to determine landing order

- Landing
    - Landing clearance
    - Taxi clearance
    - Docking clearance
- Post-Flight
    - Close-out flight plan


TERRAIN
WEATHER
FLIGHT PLAN ENTERER
TERRAIN SURVEILLANCE DATA
FLIGHT PLANS
SITE WEATHER SURV DATA
WEATHER AGENCY
GENERIC AVIATION AGENCY
GAA DATA (FIXES, BOUNDARIES, OBSTACLES)
WEATHER REPORTS
ATC PROCEDURES
GENERIC ATC
STATUS/EVENTS REPORTS
ATC ADMIN AND SUPPORT
GAA REQUIRED REPORTS
BEGIN/END SIMULATION REQUEST
GOVERNMENT
SPECIAL INTEREST REPORTS
COMMUNICATION
AIRCRAFT
AIR TRAFFIC
TRAFFIC SURVEILLANCE DATA
NON-CLOSED FLIGHT PLANS
GROUND TRAFFIC
SEARCH AND RESCUE
0.0 ATC CONTEXT
Created by: JOANNEP
Revised by: conleym
Date changed: 18-JAN-90

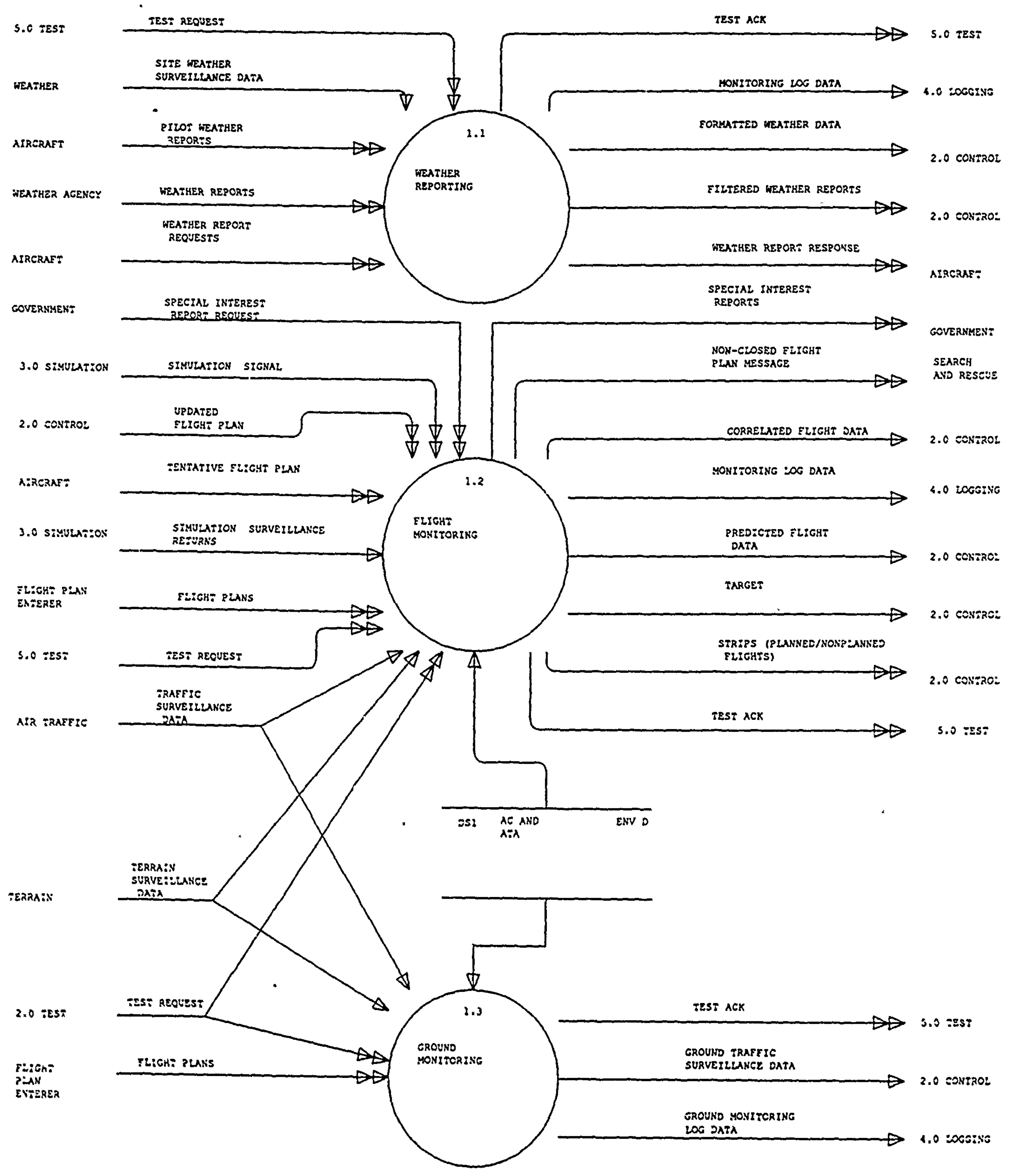


0.0 ATC Level 1
Created by: leshok
Revised by: conleyn
Date changed: 22-NOV-89



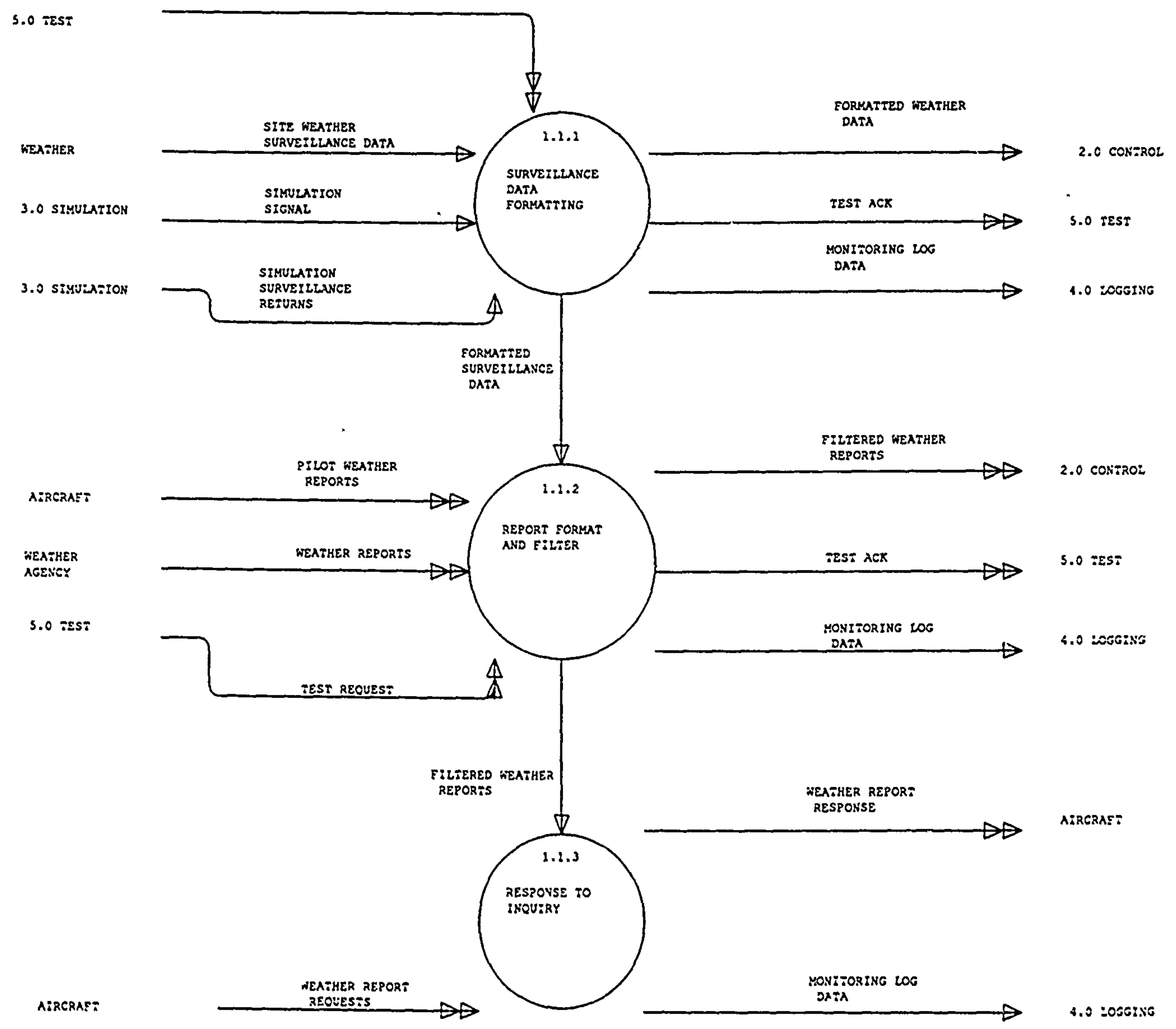
1.0 View From Monitoring
Created by: leshok
Revised by: barb
Date changed: 06-NOV-89

1.0 MONITORING



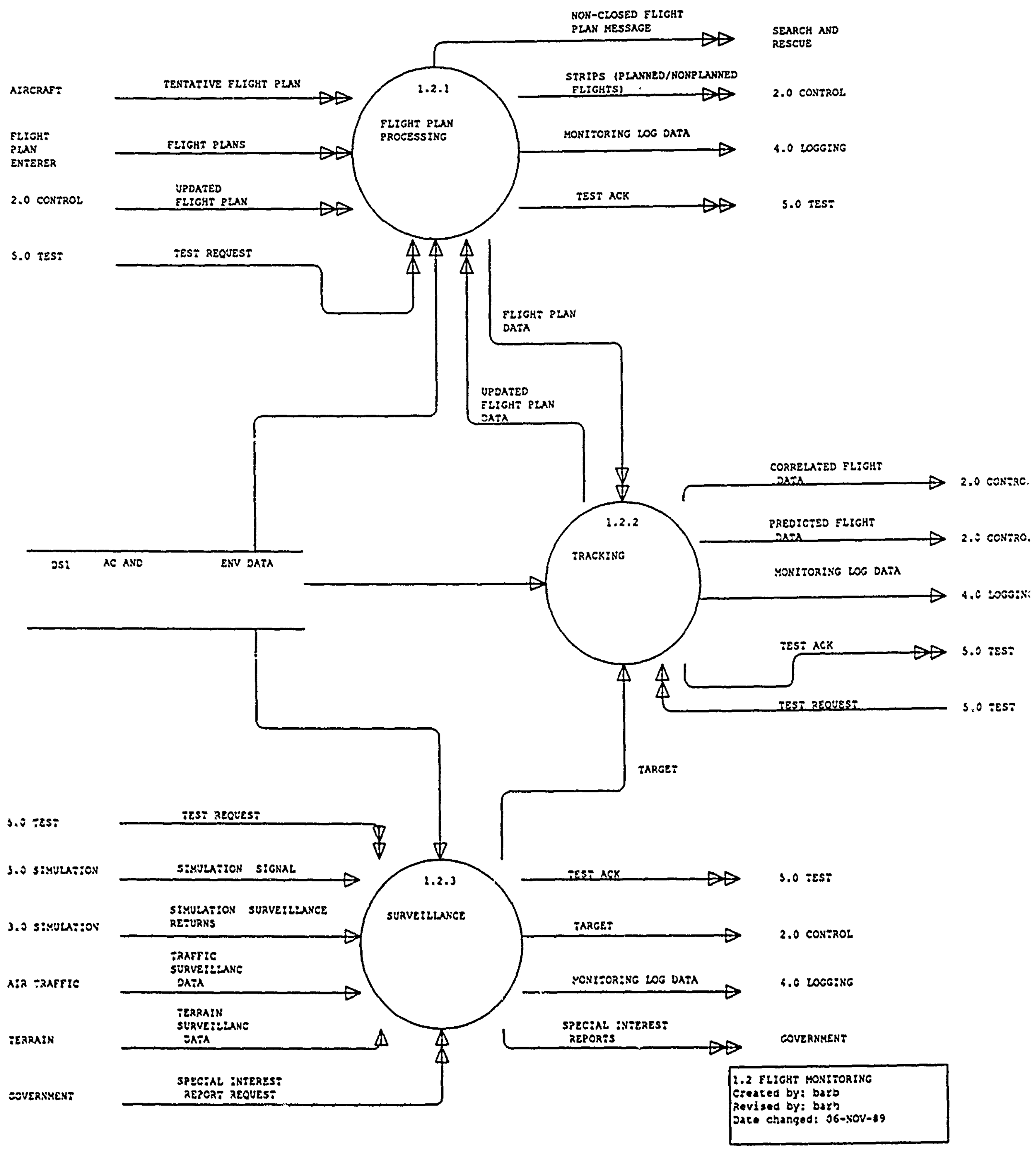
1.0 MONITORING
Created by: barb
Revised by: barb
Date changed: 06-NOV-89

1.1 WEATHER REPORTING



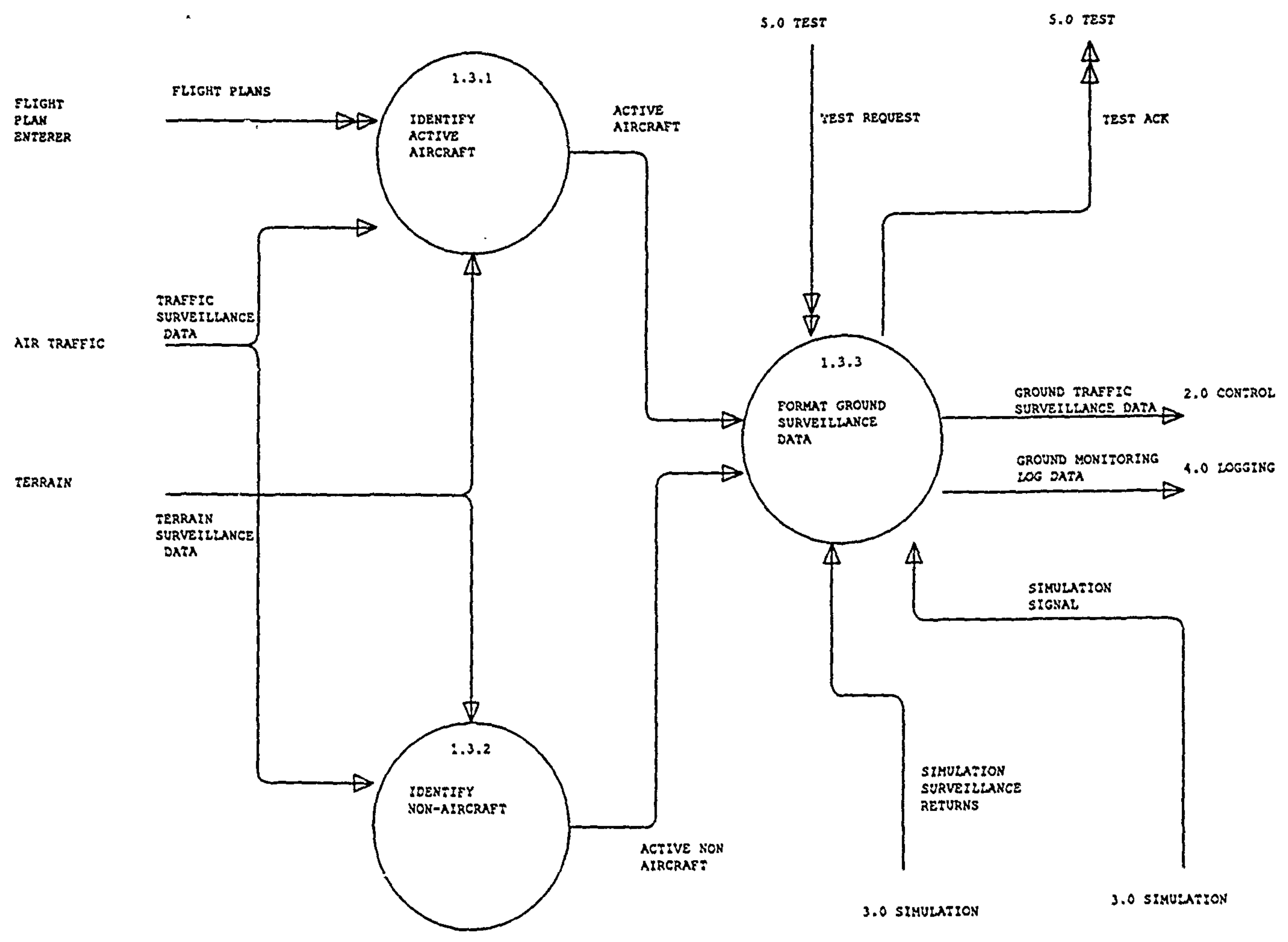
1.1 WEATHER REPORTING
Created by: barb
Revised by: conleym
Date changed: 16-JAN-90

1.2 FLIGHT MONITORING

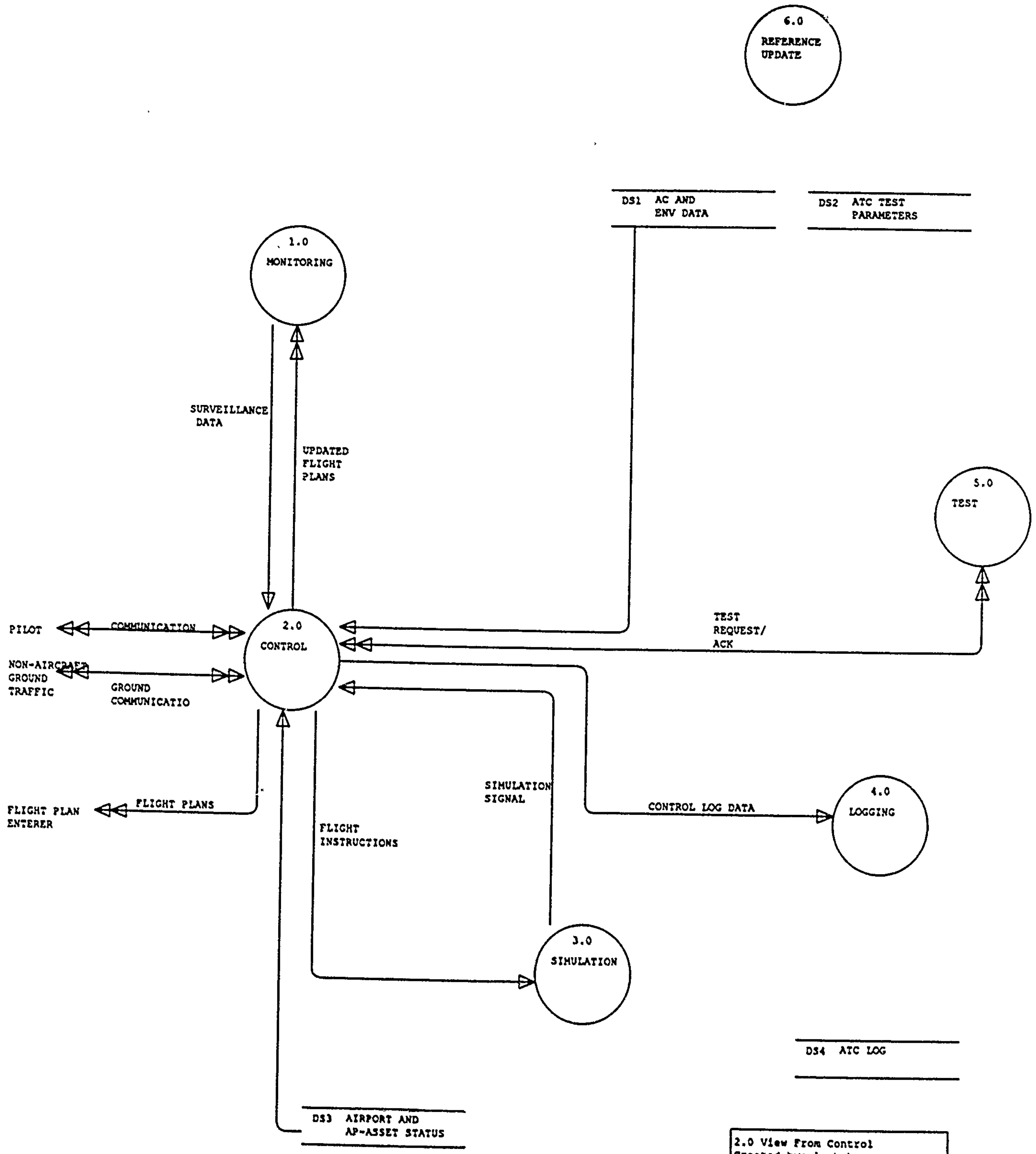


1.2 FLIGHT MONITORING
Created by: barb
Revised by: barb
Date changed: 06-NOV-89

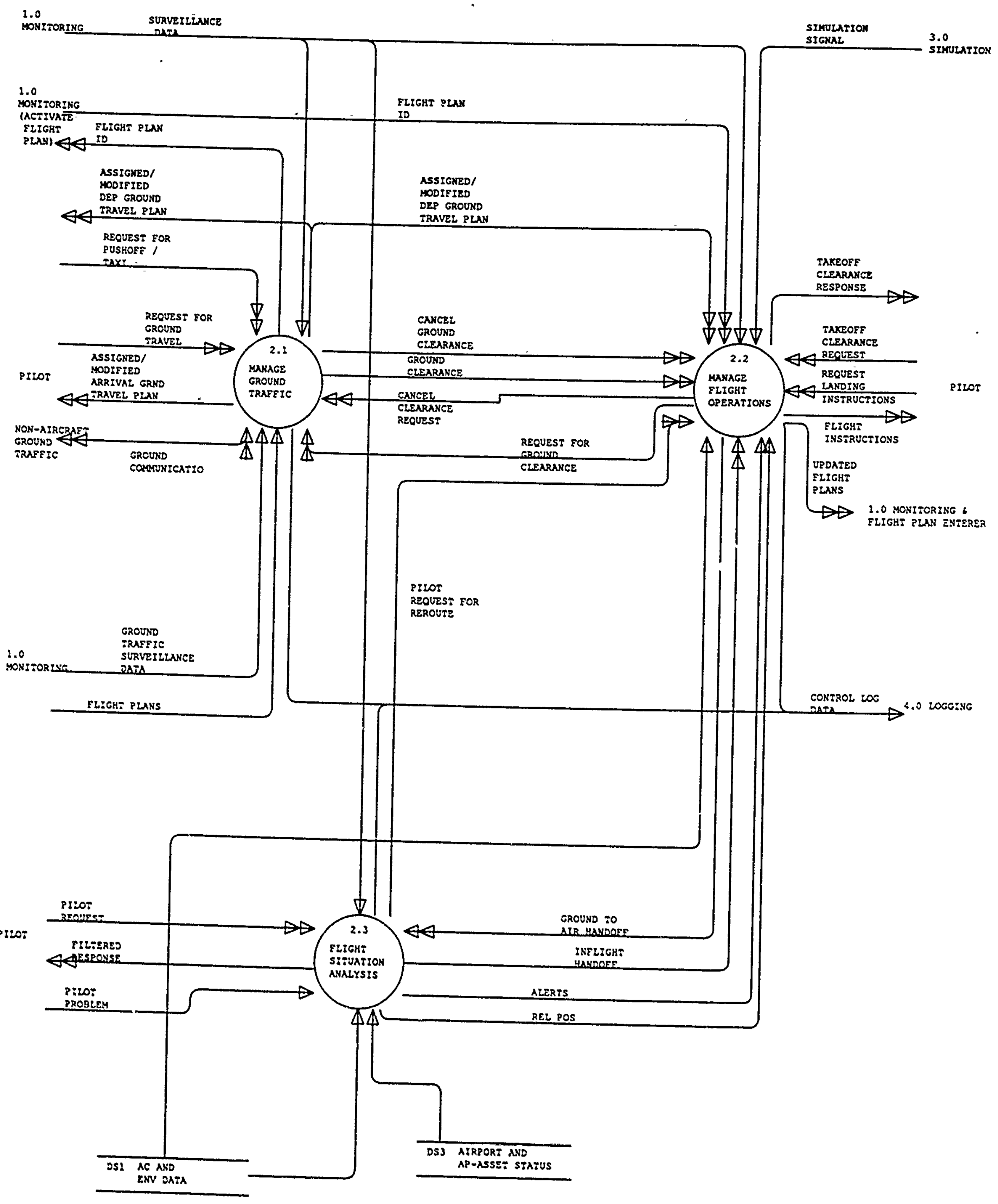
1.3 GROUND MONITORING



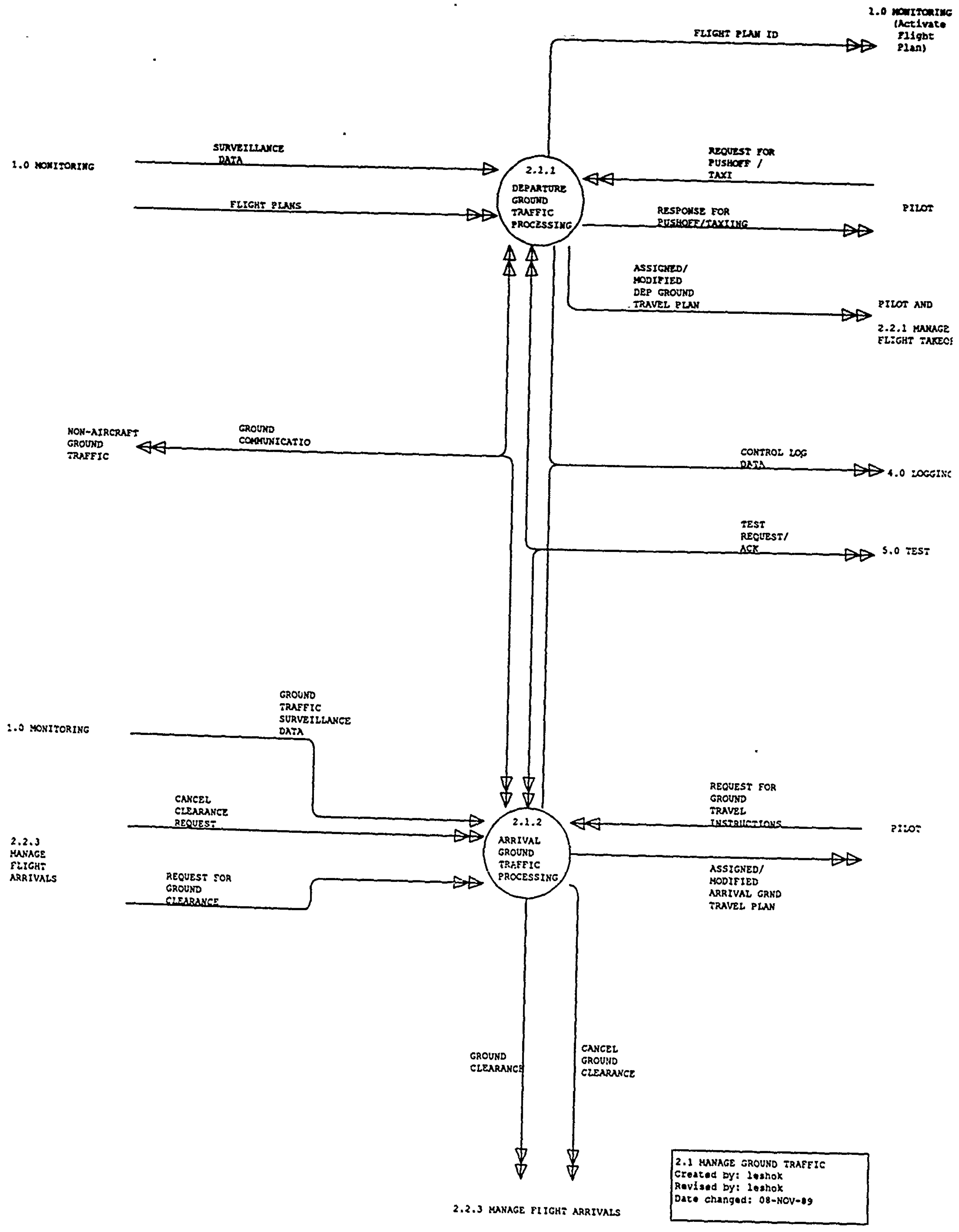
1.3 GROUND MONITORING
Created by: barb
Revised by: barb
Date changed: 06-NOV-89

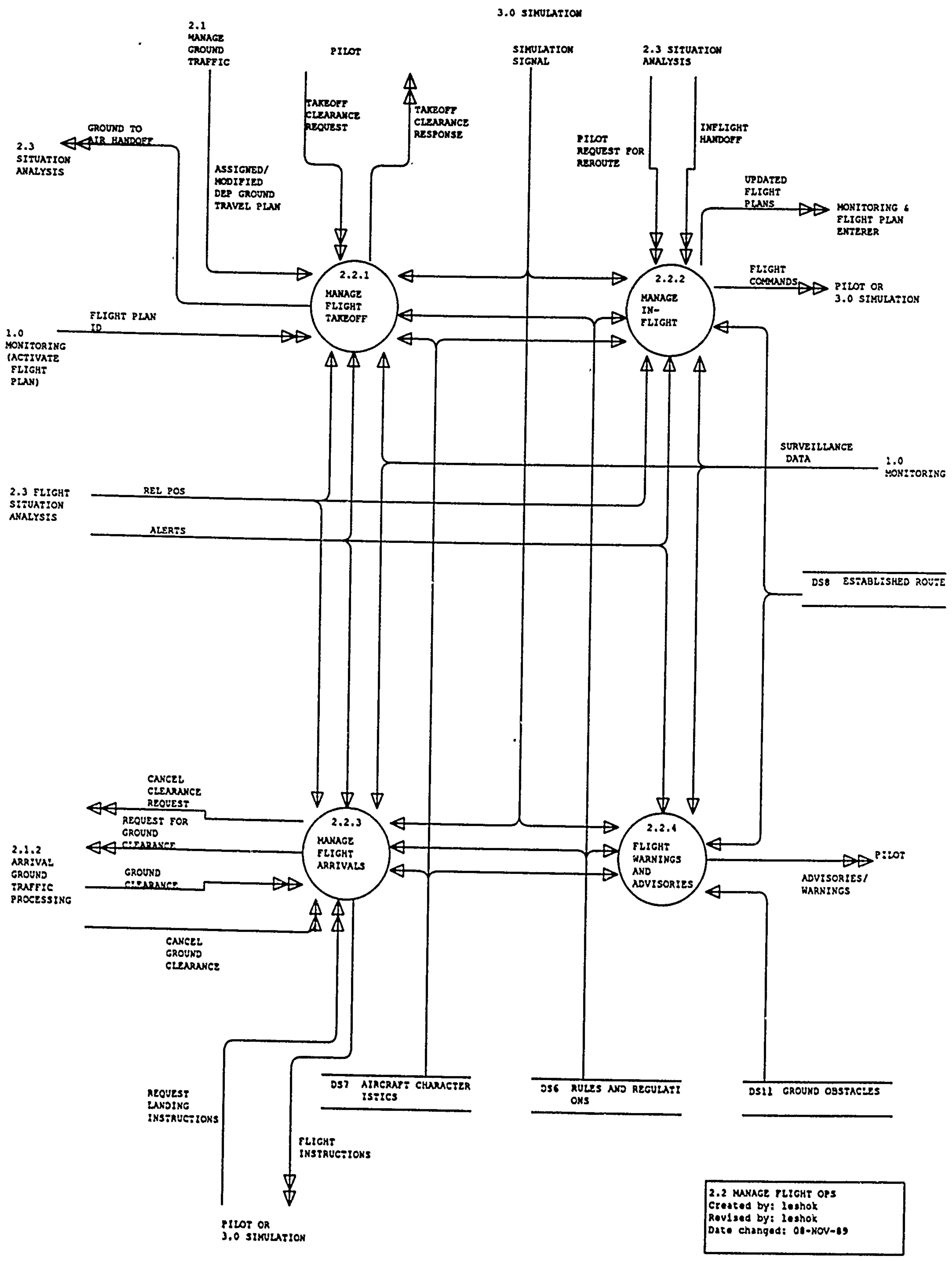


2.0 View From Control
Created by: leshok
Revised by: leshok
Date changed: 08-NOV-89

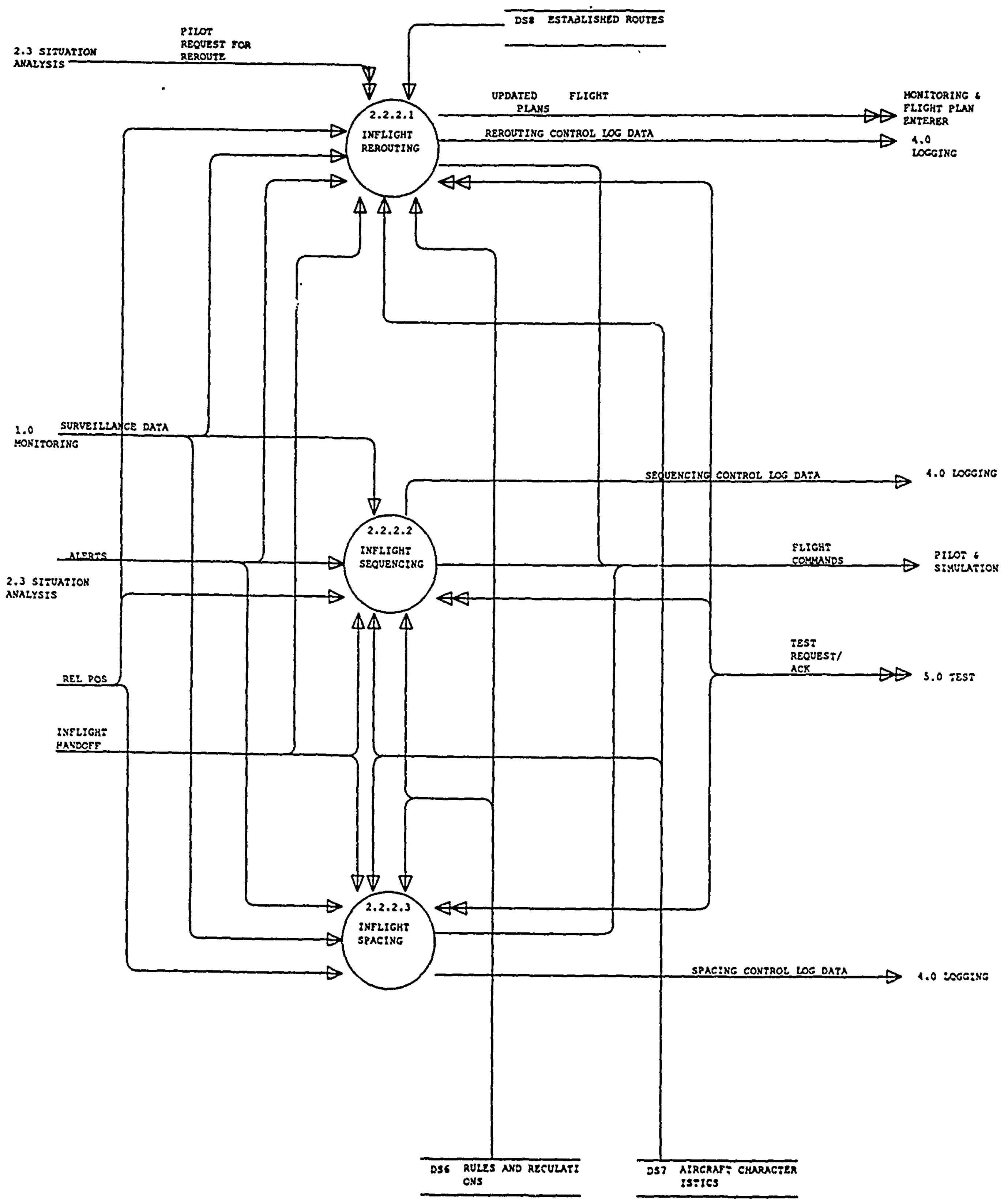

1.0 MONITORING
SURVEILLANCE DATA
SIMULATION SIGNAL
3.0 SIMULATION
1.0 MONITORING (ACTIVATE FLIGHT PLAN)
FLIGHT PLAN ID
FLIGHT PLAN ID
ASSIGNED/ MODIFIED DEP GROUND TRAVEL PLAN
ASSIGNED/ MODIFIED DEP GROUND TRAVEL PLAN
REQUEST FOR PUSHOFF / TAXI
TAKEOFF CLEARANCE RESPONSE
REQUEST FOR GROUND TRAVEL
CANCEL GROUND CLEARANCE
TAKEOFF CLEARANCE REQUEST
2.1 MANAGE GROUND TRAFFIC
2.2 MANAGE FLIGHT OPERATIONS
PILOT
ASSIGNED/ MODIFIED ARRIVAL GRND TRAVEL PLAN
GROUND CLEARANCE
REQUEST LANDING INSTRUCTIONS
PILOT
CANCEL CLEARANCE REQUEST
NON-AIRCRAFT GROUND TRAFFIC
GROUND COMMUNICATIO
REQUEST FOR GROUND CLEARANCE
FLIGHT INSTRUCTIONS
UPDATED FLIGHT PLANS
1.0 MONITORING & FLIGHT PLAN ENTERER
PILOT REQUEST FOR REROUTE
1.0 MONITORING
GROUND TRAFFIC SURVEILLANCE DATA
FLIGHT PLANS
CONTROL LOG DATA
4.0 LOGGING
PILOT REQUEST
PILOT
2.3 FLIGHT SITUATION ANALYSIS
GROUND TO AIR HANDOFF
FILTERED RESPONSE
INFLIGHT HANDOFF
PILOT PROBLEM
ALERTS
REL POS
DS1 AC AND ENV DATA
DS3 AIRPORT AND AP-ASSET STATUS

2.0 CONTROL
Created by: JOANNEP
Revised by: leshok
Date changed: 08-NOV-89

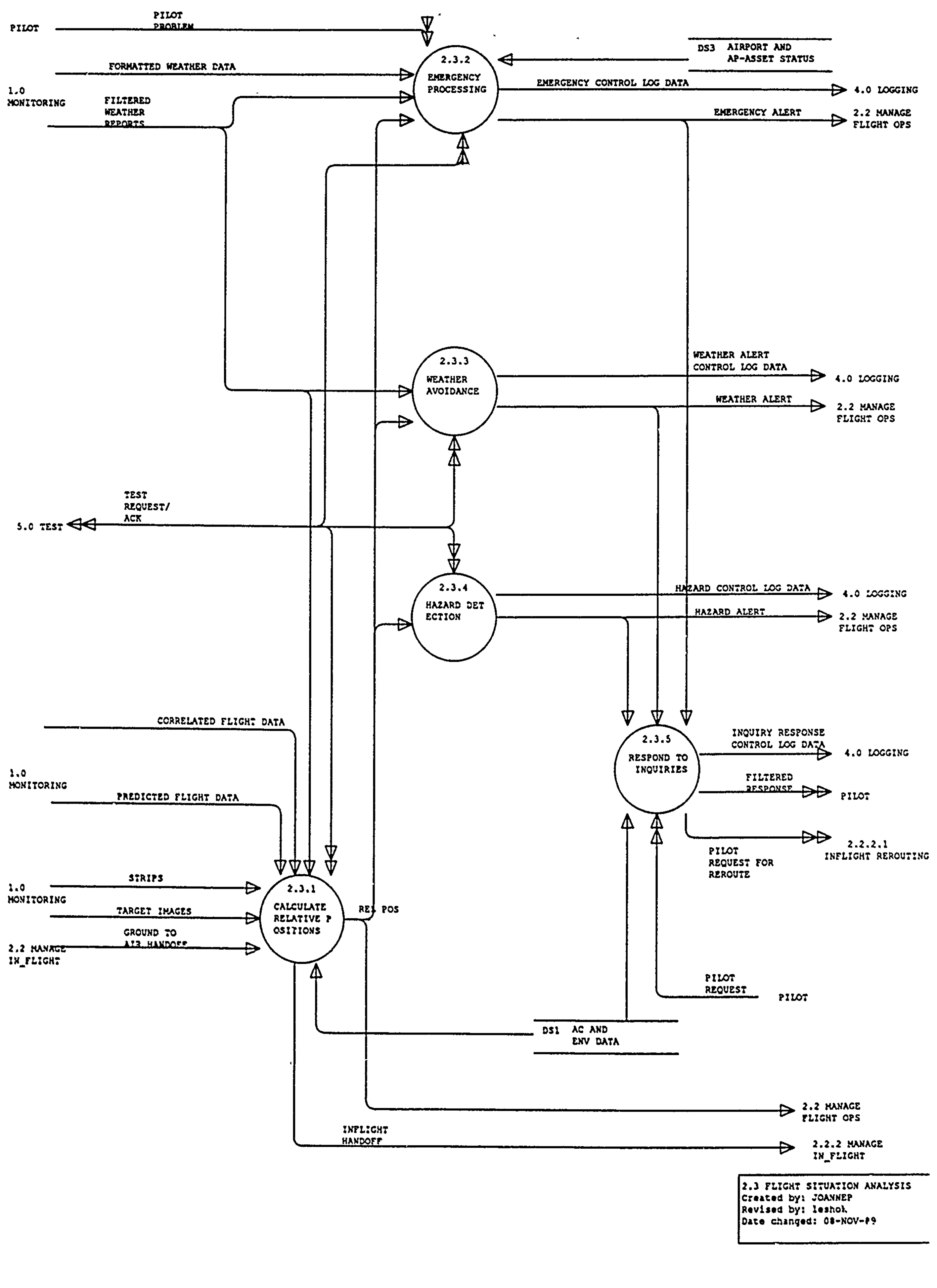

1.0 MONITORING (Activate Flight Plan)
FLIGHT PLAN ID
1.0 MONITORING
SURVEILLANCE DATA
FLIGHT PLANS
2.1.1 DEPARTURE GROUND TRAFFIC PROCESSING
REQUEST FOR PUSHOFF / TAXI
PILOT
RESPONSE FOR PUSHOFF/TAXIING
ASSIGNED/ MODIFIED DEP GROUND TRAVEL PLAN
PILOT AND
2.2.1 MANAGE FLIGHT TAKEO
NON-AIRCRAFT GROUND TRAFFIC
GROUND COMMUNICATIO
CONTROL LOG DATA
4.0 LOGGING
TEST REQUEST/ ACK
5.0 TEST
1.0 MONITORING
GROUND TRAFFIC SURVEILLANCE DATA
CANCEL CLEARANCE REQUEST
2.2.3 MANAGE FLIGHT ARRIVALS
REQUEST FOR GROUND CLEARANCE
2.1.2 ARRIVAL GROUND TRAFFIC PROCESSING
REQUEST FOR GROUND TRAVEL INSTRUCTIONS
PILOT
ASSIGNED/ MODIFIED ARRIVAL GRND TRAVEL PLAN
GROUND CLEARANCE
CANCEL GROUND CLEARANCE
2.2.3 MANAGE FLIGHT ARRIVALS
2.1 MANAGE GROUND TRAFFIC
Created by: leshok
Revised by: leshok
Date changed: 08-NOV-89


3.0 SIMULATION
2.1 MANAGE GROUND TRAFFIC
PILOT
SIMULATION SIGNAL
2.3 SITUATION ANALYSIS
TAKEOFF CLEARANCE REQUEST
TAKEOFF CLEARANCE RESPONSE
GROUND TO AIR HANDOFF
2.3 SITUATION ANALYSIS
ASSIGNED/ MODIFIED DEP GROUND TRAVEL PLAN
PILOT REQUEST FOR REROUTE
INFLIGHT HANDOFF
UPDATED FLIGHT PLANS
MONITORING & FLIGHT PLAN ENTERER
2.2.1 MANAGE FLIGHT TAKEOFF
2.2.2 MANAGE IN-FLIGHT
FLIGHT COMMANDS
PILOT OR 3.0 SIMULATION
FLIGHT PLAN ID
1.0 MONITORING (ACTIVATE FLIGHT PLAN)
SURVEILLANCE DATA
1.0 MONITORING
2.3 FLIGHT SITUATION ANALYSIS
REL POS
ALERTS
DS8 ESTABLISHED ROUTE
CANCEL CLEARANCE REQUEST
REQUEST FOR GROUND CLEARANCE
2.1.2 ARRIVAL GROUND TRAFFIC PROCESSING
GROUND CLEARANCE
2.2.3 MANAGE FLIGHT ARRIVALS
2.2.4 FLIGHT WARNINGS AND ADVISORIES
PILOT
ADVISORIES/ WARNINGS
CANCEL GROUND CLEARANCE
REQUEST LANDING INSTRUCTIONS
FLIGHT INSTRUCTIONS
DS7 AIRCRAFT CHARACTERISTICS
DS6 RULES AND REGULATIONS
DS11 GROUND OBSTACLES
PILOT OR 3.0 SIMULATION

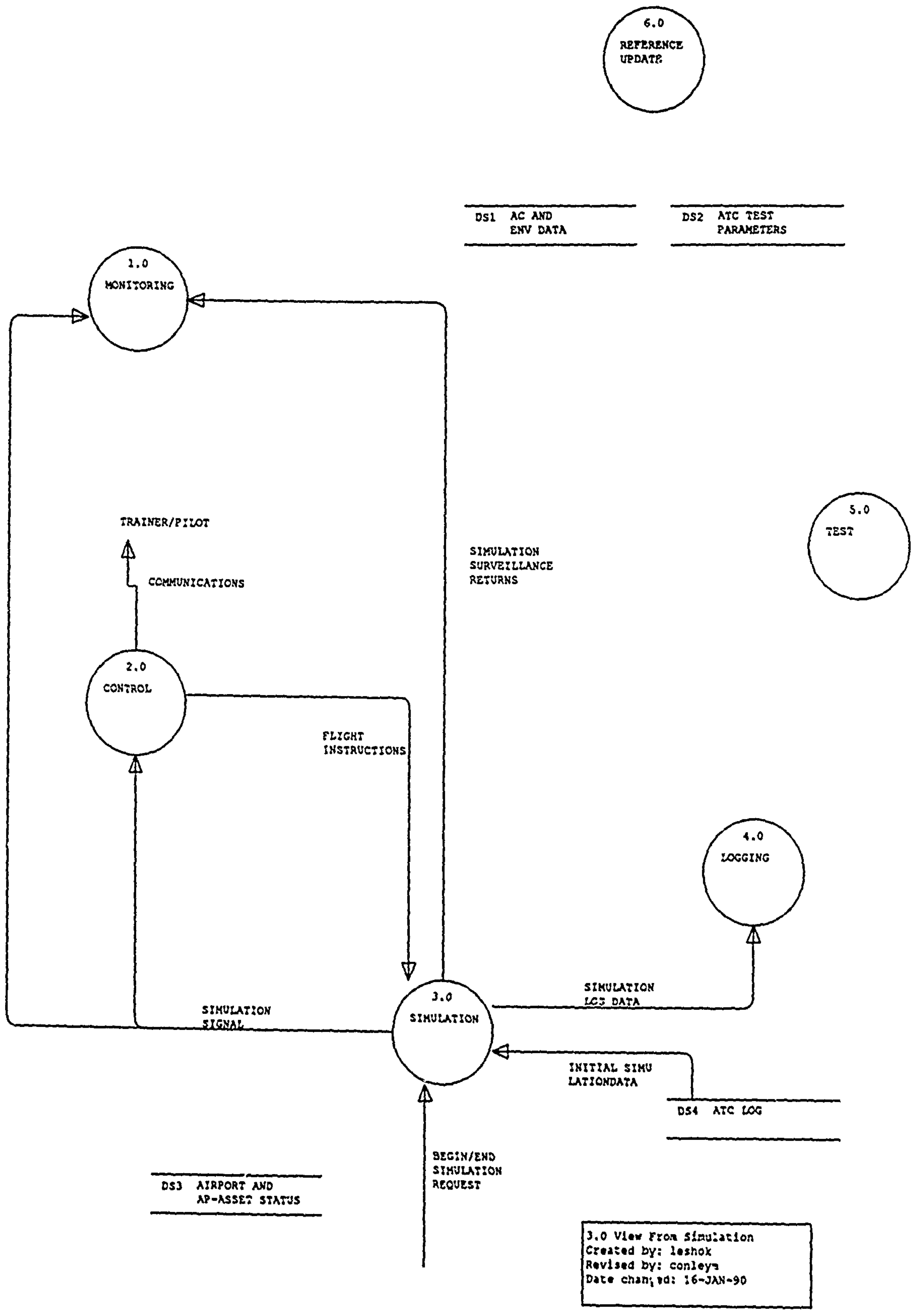
2.2 MANAGE FLIGHT OPS
Created by: leshok
Revised by: leshok
Date changed: 08-NOV-89



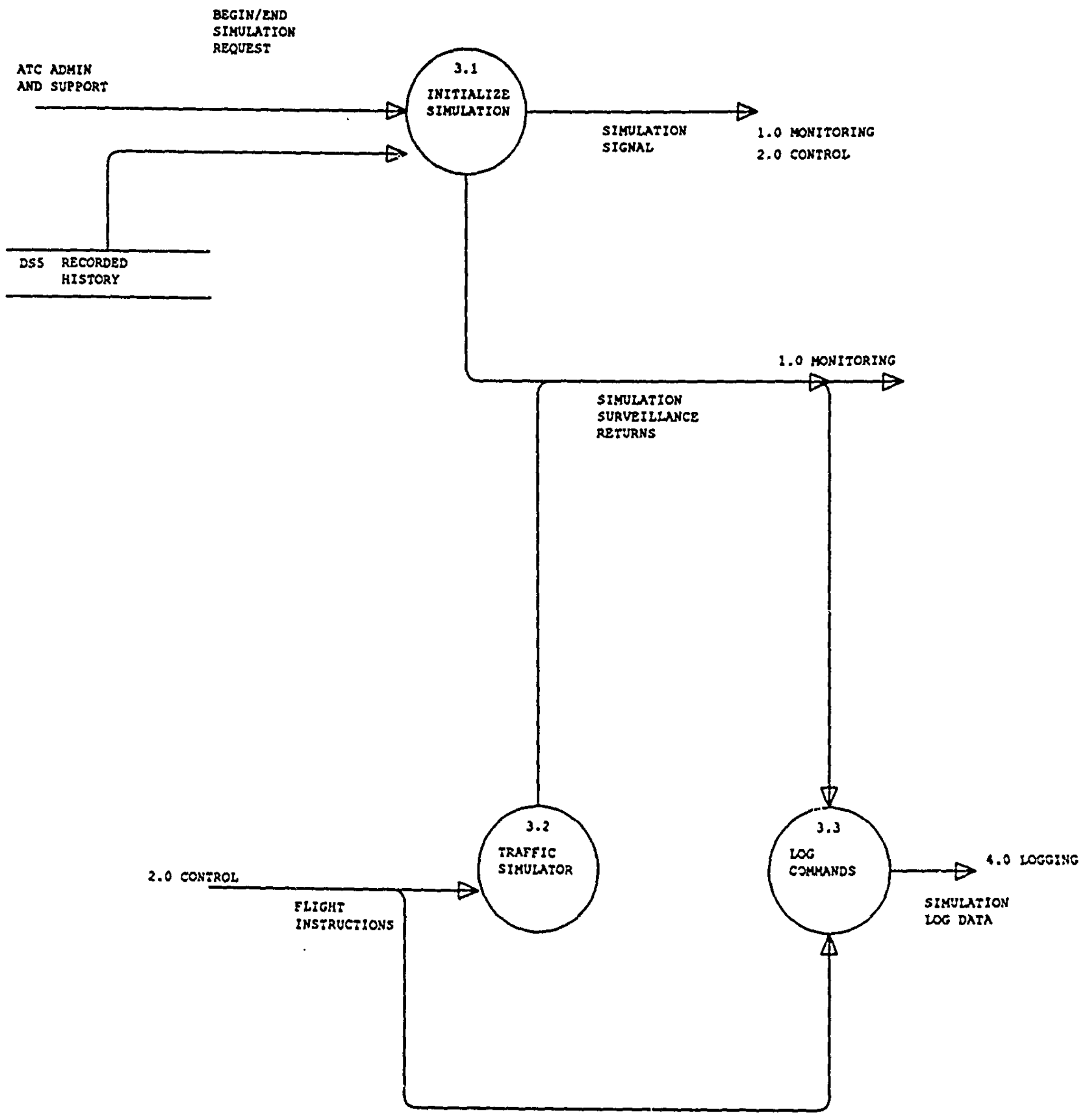
2.2.2 MANAGE INFLIGHT
Created by: JOANNEP
Revised by: leshok
Date changed: 08-NOV-89



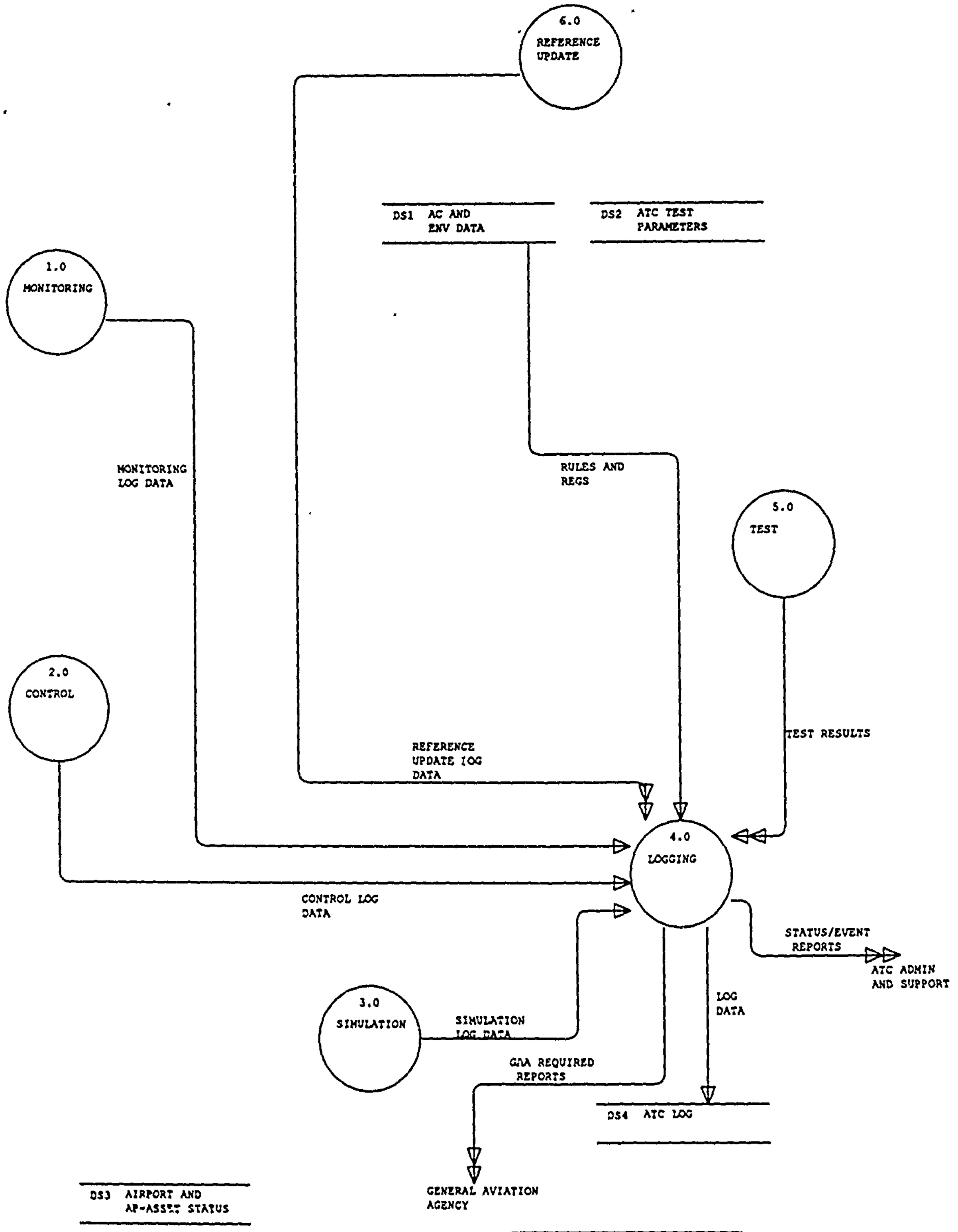
2.3 FLIGHT SITUATION ANALYSIS
Created by: JOANNEP
Revised by: leshok
Date changed: 08-NOV-89



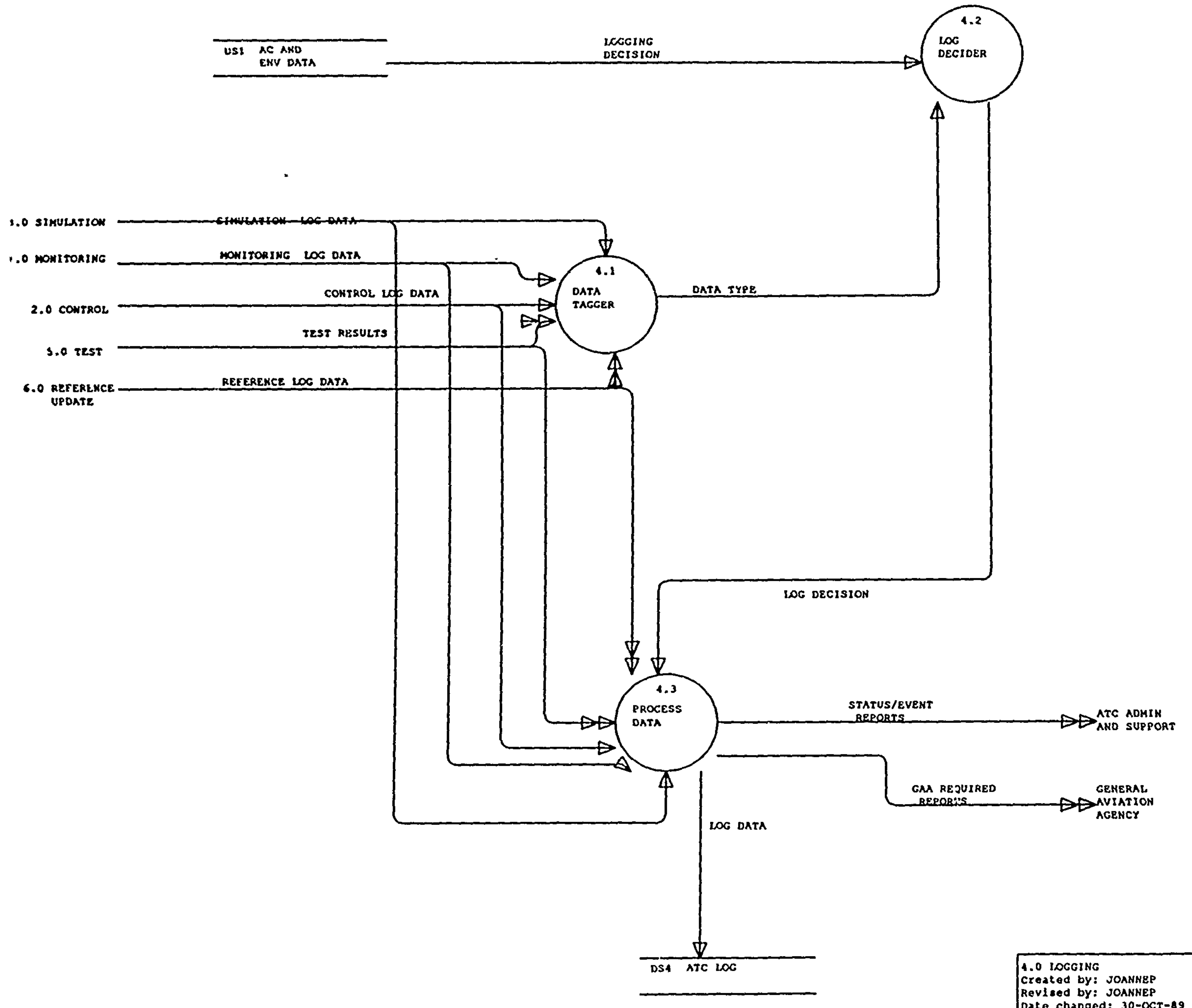
3.0 View From Simulation
Created by: leshok
Revised by: conleym
Date changed: 16-JAN-90



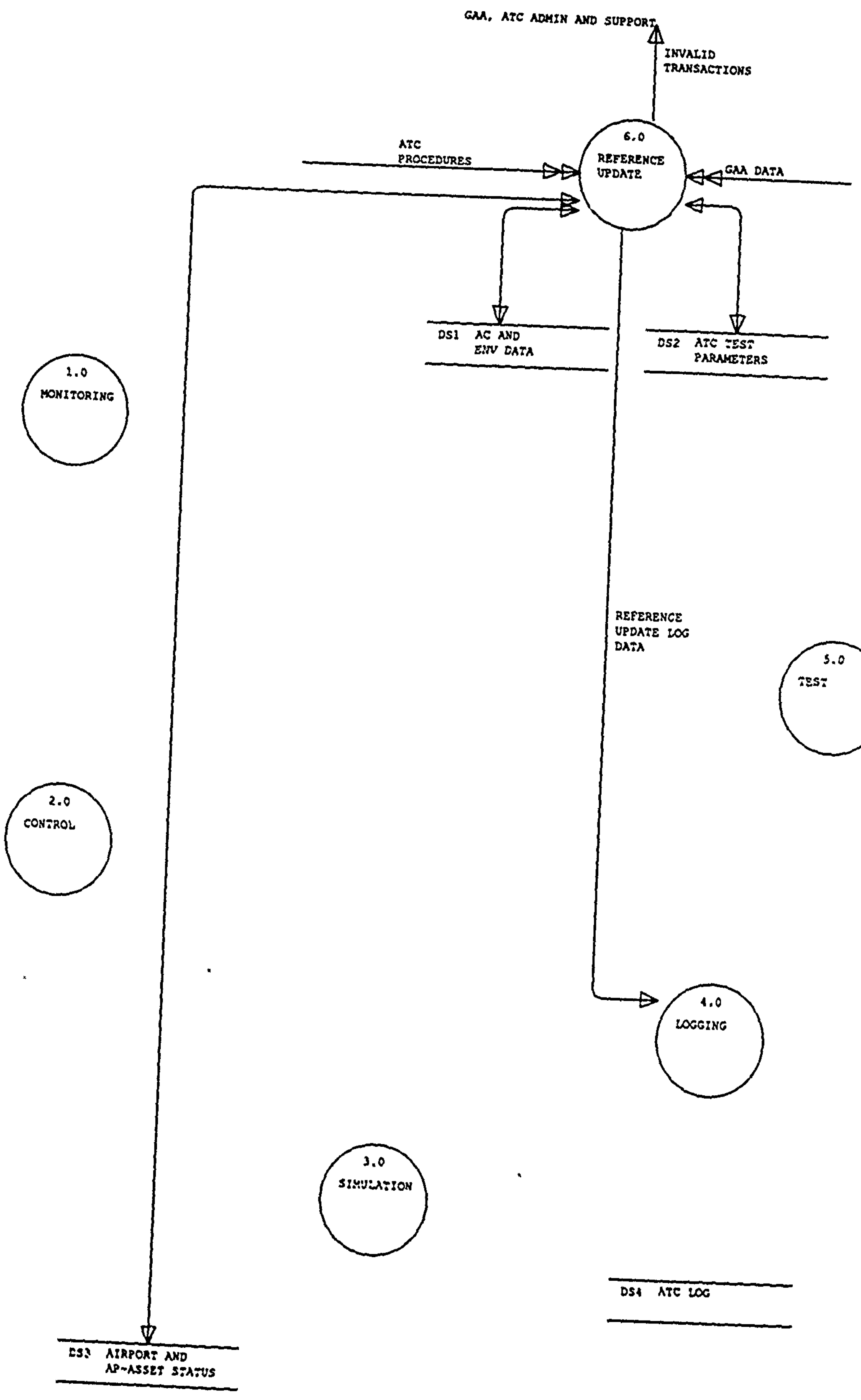
3.0 SIMULATION
Created by: bob
Revised by: conleym
Date changed: 16-JAN-90



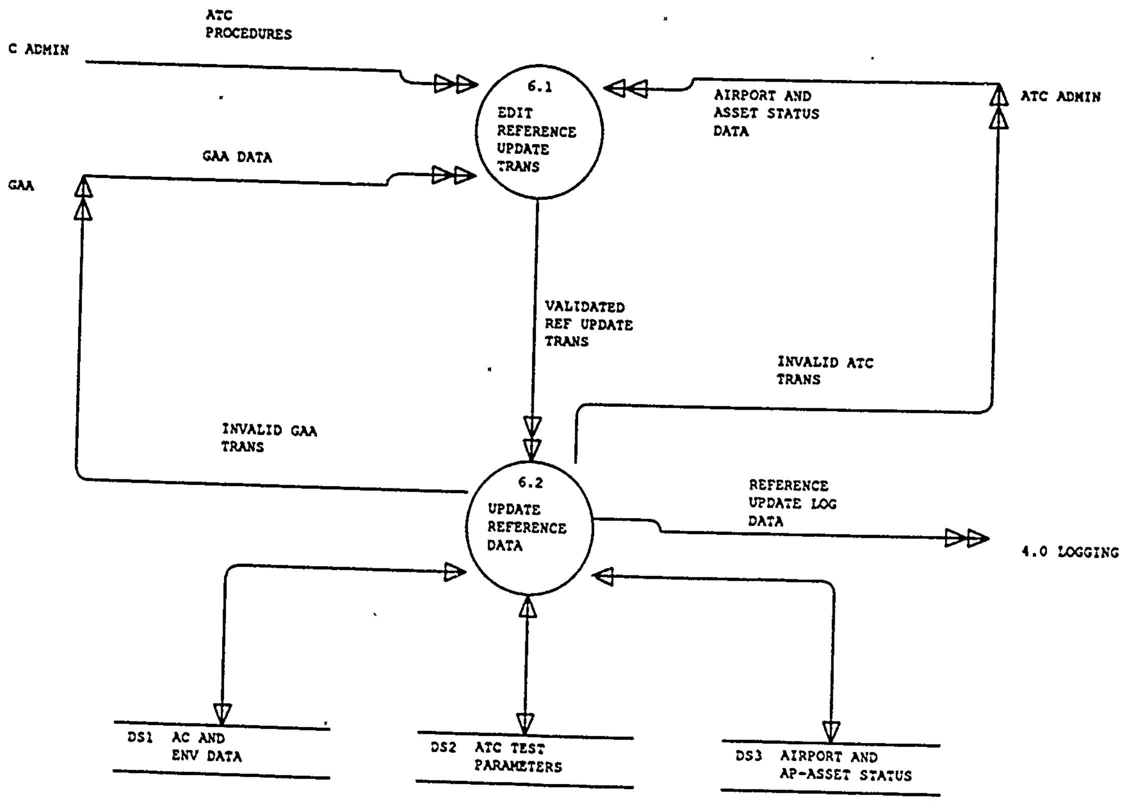
4.0 View From Logging
Created by: leshok
Revised by: conleym
Date changed: 16-JAN-90



4.0 LOGGING
Created by: JOANNEP
Revised by: JOANNEP
Date changed: 30-OCT-89



6.0 View From Reference Update
Created by: leshok
Revised by: bob
Date changed: 27-OCT-89



6.0 REFERENCE UPDATE
Created by: ettw
Revised by: bob
Date changed: 27-OCT-89

```
DATE: 16-JAN-90                 *** ALL PROCESSES FOR ATC MODEL ***                    PAGE     1
TIME: 11:15                                                                              EXCEL/RTS

Alternate Name                   Short Descrip.                                           Last Modify Date
-------------------------------- -------------------------------------------------------- ----------------
1.0 MONITORING                                                                                  891024

1.1 WEATHER REPORTING            Performs the gathering of surveillance data regarding the      891027
                                 weather.

1.1.1 SURVEILLANCE DATA FORMATTI Performs gathering and formatting of surveillance data         891027

1.1.3 REPORT_FORMAT_AND_FILTER   Gathers weather data in the form of weather reports, pilot     891027
                                 weather reports and surv. data and filters it for CONTROL

1.1.3 RESPONSE TO INQUIRY        Responds to pilots request for weather data                    891027

1.2 FLIGHT MONITORING            Performs gathering of surveillance data of aircraft as they    891027
                                 are in-flight.

1.2.1 FLIGHT_PLAN_PROCESSING     Accepts flight plans and revisions and sends abbrev flight     891027
                                 plans to CONTROL and active flight plan info to TRACKING

1.2.2 TRACKING                   Tracks targets as they move and correlates them to flight      891027
                                 plans if available

1.2.3 SURVEILLANCE               Performs gathering of flight surveillance data for TRACKING    891027
                                 and CONTROL

1.3 GROUND MONITORING            Performs the gathering of surveillance data at the             891027
                                 airports, Watches the aircraft on the ground.

1.3.1 IDENTIFY_ACTIVE_AIRCRAFT   Identifies active (gate to runway) aircraft at the airport.    891027

1.3.2 IDENTIFY_NON-AIRCRAFT      Identifies other active traffic, besides aircraft, at the      891027
                                 airport

1.3.3 FORMAT_GROUND_SURVEILLANCE Combines the airport traffic information on both the active   891027
                                 aircraft and the active non-aircraft for output to CONTROL

2.0 CONTROL                      CONTROL receives data about the environment from               891027
                                 MONITORING, and performs analysis and issues control inst.

2.1 MANAGE GROUND TRAFFIC        Manages ground traffic for departures and arrivals.            891026

DEPARTURE GROUND TRAFFIC PROCESS Manages ground traffic for departures.  Processes TAXIING      891027
                                 requests and provides ground travel plans.

ARRIVAL GROUND TRAFFIC PROCESSIN Manages ground traffic for arrivals.  Processes GROUND         891027
                                 CLEARANCE requests and provides ground travel plans.

2.2 MANAGE FLIGHT OPERATIONS     Manages aircraft from the moment the aircraft leaves the       891026
                                 ground until it once again reaches the ground.

MANAGE_FLIGHT_TAKEOFF            Process departures, from the time the AIRCRAFT reaches the     891027
                                 takeoff runway.  Sends TAKEOFF CLEARANCE to PILOT.
```

Alternate Name	Short Descrip.	Last Modify Date
MANAGE_IN_FLIGHT	Manages AIRCRAFT in flight (after leaving ground up to GROUND CLEARANCE request). Provides FLIGHT CMDS to PILOT.	891027
2.2.2.1 INFLIGHT REROUTING	Rerouting (ie. changing destination or way points) of an aircraft. Instructions to PILOT, and updated flight plan.	891027
2.2.2.2 INFLIGHT SEQUENCING	Sequences (orders) planes for arrival or through points where routes may intersect.	891027
2.2.2.3 INFLIGHT SPACING	Spaces aircraft in order to keep them the distance apart as indicated in the RULES AND REGS.	891027
2.2.2.4 WARNINGS AND ADVISORIES		891024
MANAGE_FLIGHT_ARRIVALS	Manages flight arrivals. Requests GROUND CLEARANCE and provides landing instructions to PILOT or SIMULATION.	891027
FLIGHT_WARNINGS_&_ADVISORIES	Provides WARNINGS AND ADVISORIES to PILOT. Provided info from alerts generated in 2.3 SITUATION ANALYSIS.	891027
2.3 FLIGHT SITUATION ANALYSIS	Using SURVEILLANCE DATA from MONITORING, calculates rel pos., determines alerts, and responds to pilot inquiries.	891026
2.3.1 CALCULATE RELATIVE POSITIO	Calculates the distances between aircraft and other aircraft, weather, and terrain obstacles.	891026
2.3.2 EMERGENCY PROCESSING	Analysis input, checking for emergencies caused by pilot/aircraft problems, closed airports/runways, etc.	891026
2.3.3 WEATHER AVOIDANCE	Analyses input, checking for weather hazards and produces alerts based on the weather situation.	891026
2.3.4 HAZARD DETECTION	Analyses input, checking for hazards such as conflict alerts (2 aircraft too close), low altitude alerts, etc.	891026
2.3.5 RESPOND TO INQUIRIES	Receives pilot inquiries about the weather/traffic/terrain and responds.	891026
3.0 SIMULATION	SIMULATION provides support for Controller training and the testing of new ATC capabilities before field installation.	891027
3.1 INITIALIZE SIMULATION	Uses RECORDED HISTORY to initialize the environment and conditions in the simulator.	891027
3.2 TRAFFIC SIMULATOR	Uses controller commands and INITIAL SIMULATION SURVEILLANCE RETURNS to calculate new traffic positions.	891027
3.3 LOG COMMANDS	Logs everything that needs to be recorded in the ATC system log.	891027
4.0 LOGGING		891024

```
DATE: 16-JAN-90              *** ALL PROCESSES FOR ATC MODEL ***                   PAGE     3
TIME: 11:15                                                                          EXCEL/RTS

Alternate Name                    Short Descrip.                                          Last Modify Date
--------------------------------- ------------------------------------------------------- ----------------
4.1 DATA TAGGER                   The process DATA TAGGER identifies the type of log data           891027
                                  and outputs a DATA TYPE tag for the data.

4.2 LOG DECIDER                   The process which based on the LOGGING DECISION determines        891027
                                  whether log data of an identified type is to be logged.

4.3 PROCESS DATA                  PROCESS DATA filters the log data based on the LOG DECISION       891027
                                  and formats the data, and writes the data to the ATC_LOG.

5.0 TEST                                                                                            891024

5.1 HEALTH AND WELLBEING TEST     Performs general health tests on the online system.The test       891027
                                  type and frequency are based on the ASSET TEST RQMTS.

5.1.1 INITIATE_TEST               Send health and wellbeing test request to MONITORING or           891027
                                  CONTROL and inform EVALUATE TEST RESULTS of test request.

5.1.2 EVALUATE_TEST_RESULT        Evaluate the test results upon test completion,send requests      891027
                                  for special tests to SPECIAL TEST and log the test results.

5.2 SPECIAL TESTS                 This process performs online special tests as a result of a       891027
                                  log data anomaly or a health and wellbeing test result.

5.2.1 INITIATE TEST               Send special test requests to MONITORING or CONTROL and           891027
                                  inform EVALUATE TEST RESULTS of the test request.

5.2.2 EVALUATE TEST RESULTS       Evaluate the test results upon test completion and log the        891027
                                  test results.

5.3 EXAMINE LOG DATA              This process examines the ATC_LOG data to identify                891027
                                  anomalies that require further testing of the system.

5.3.1 IDENTIFY ANOMALIES          Examines the ATC_LOG data for anomalies using the ATC TEST        891027
                                  PARAMETERS to provide equipment and operational data.

5.3.2 IDENTIFY SPECIAL TEST       Based on the input ANOMALY and the special test                   891027
                                  requirements in ASSET TEST RQMNTS, request a special test.

6.0 REFERENCE UPDATE              REFERENCE UPDATE provides data management services for all        891027
                                  adaptation data, operational procedures, rules & regs, etc.

6.1 EDIT REFERENCE UPDATE TRANS   Data for updating the ATC Reference data is given a               891025
                                  validity check and passed to the update procedure.

6.2 UPDATE REFERENCE DATA         The validated transactions are processed and the REFERENCE        891025
                                  DATA data stores are updated.

ATC System                                                                                          891024
```

DATE: 16-JAN-90
TIME: 10:57

*** ALL DATA FLOWS FOR ATC MODEL ***

PAGE 1
EXCEL/RTS

Name	Alternate Name	Short Descrip.	Last Modify Date
H0025	ACTIVE_AIRCRAFT	Aircraft which are active in ground operations, from gate to runway.	891027
H0026	ACTIVE_NON-AIRCRAFT	Other ground traffic, such as trucks, which must be identified and avoided during aircraft ground travel	891027
K0049	ADVISORIES_WARNINGS	Warnings and advisories given to the PILOT. Generated from the ALERTS passed to 2.2.4 FLIGHT WARNINGS AND ADVISORIES.	891027
E0003	AIRPORT_AND_ASSET_STATUS_DATA	Dynamic data describing the airport status and the availability or non-availability of critical airport assets.	891025
K0046	ALERTS	Alerts generated by FLIGHT SITUATION ANALYSIS, consists of Emergency, Weather, and Hazard alerts.	891027
J0019	ANOMALY	A discrepancy in the ATC_LOG data.	891027
K0031	ASSIGNED_MOD_ARR_GRD_TRAVEL_PLAN	Travel plan given to the PILOT by MANAGE GROUND TRAFFIC for arrivals.	891030
K0036	ASSIGNED_MOD_DEP_GRD_TRAVEL_PLAN	Travel plan given to the PILOT and MANAGE FLIGHT OPERATIONS by MANAGE GROUND TRAFFIC for departures.	891030
E0001	ATC_PROCEDURES	Operating procedures for ATC site maintenance and testing of site assets.	891025
H0030	BEGIN/END_SIMULATION_REQUEST	Request coming in from the Air traffic administration to have the ATC system put in simulation mode.	891027
K0038	CANCEL_CLEARANCE_REQUEST	Cancel the CLEARANCE REQUEST issued for flight indicated by the FLIGHT PLAN ID.	891030
K0041	CANCEL_GROUND_CLEARANCE	Cancels GROUND CLEARANCE given for flight identified by FLIGHT PLAN ID.	891030
S0003	COMMUNICATIONS	Communications that go from SIMULATION to CONTROL and back. May include Pilot requests.	891027
K0018	COMMUNICATIONS	Communication to/from the PILOT. Includes requests to takeoff/land, navigation inst., inquiries, warnings, etc.	891027
K0009	CONTROL_LOG_DATA	Data generated by CONTROL to be logged. Includes alerts and communication to the AIRCRAFT.	891030
H0015	CORRELATED_FLIGHT_DATA	Surveillance Data and Flight Plans that have been matched. Today shows which flight plan goes with which radar blip.	891027
K0043	EMERGENCY_ALERT	Alerts generated by 2.3.2 EMERGENCY PROCESSING, because of PILOT PROBLEMS or some change in AIRPORT & AP ASSET STATUS.	891027
K0012	EMERGENCY_CONTROL_LOG_DATA	Data logged by 2.3.2 EMERGENCY PROCESSING. Includes ALERTS generated due to emergencies (ie. PILOT PROBLEMS).	891027

Name	Alternate Name	Short Descrip.	Last Modify Date
K0050	FILTERED_RESPONSE	Response given to PILOT by 2.3.5, RESPOND TO INQUIRIES.	891030
H0007	FILTERED_WEATHER_REPORTS	Incoming weather data, weather reports and pilot reports, that has been filtered for use by the CONTROL function	891027
K0051	FLIGHT_COMMANDS	Navigation instructions from MANAGE INFLIGHT to the PILOT.	891027
K0003	FLIGHT_INSTRUCTIONS	Flight instructions from CONTROL to the AIRCRAFT, include navigation inst., clearances, ground routes, etc.	891027
H0028	FLIGHT_INSTRUCTIONS	Commands from CONTROL telling traffic what action to perform.	891027
H0013	FLIGHT_PLANS	Itinery of a particular flight which is entered by prior to a flight commencing. Also includes beacon code, etc.	891122
H0024	FLIGHT_PLAN_DATA	Flight plan data that is required by TRACKING to perform correlation between flight plans and identified targets.	891027
K0033	FLIGHT_PLAN_ID	Indicates a specific flight plan (and therefore a specific AIRCRAFT) within the system.	891030
H0023	FORMATTED_SURVEILLANCE_DATA	Surveillance weather data that is formatted for use by the REPORT FORMAT AND FILTER function	891027
H0006	FORMATTED_WEATHER_DATA	Weather surveillance data that has been formatted for use by the CONTROL function	891027
E0002	GAA_DATA	Data provided by the <GENERIC> Aviation Agency to establish the policy, procedures and operating parameters for ATC ops.	891026
J0044	GAA_REQUIRED_REPORTS	General Aviation Agency requested reports generated from the log data.	891027
K0040	GROUND_CLEARANCE	Indicates the ground is clear for an AIRCRAFT to land. Given to 2.2.3 MANAGE FLIGHT ARRIVALS.	891030
K0057	GROUND_COMMUNICATION	Communication between CONTROL and NON_AIRCRAFT GROUND TRAFFIC, for the purpose of controlling runway traffic.	891106
H0022	GROUND_MONITORING_LOG_DATA	Logged data from the GROUND MONITORING function.	891027
K0048	GROUND_TO_AIR_HANDOFF	Indication to 2.3 SITUATION ANALYSIS that AIRCRAFT (indicated in rec.) has left the ground.	891027
H0021	GROUND_TRAFFIC_SURVEILLANCE_DATA	Surveillance data about ground traffic at an airport.	891027
K0032	HAZARD_ALERT	Using SURVEILLANCE DATA and REL POS, determines hazards such as CONFLICT ALERT and LOW ALTITUDE.	891027
K0027	HAZARD_CONTORL_LOG_DATA	Log data generated by 2.3.4 HAZARD DETECTION Includes hazard alerts.	891027

Name	Alternate Name	Short Descrip.	Last Modify Date
J0010	HEALTH_AND_WELLBEING_ACK	Sent by MONITORING and CONTROL indicating that the Health and being test was performed.It includes the test results.	891027
J0009	HEALTH_AND_WELLBEING_REQUEST	A request to MONITORING or CONTROL that a Health and Wellbeing test be performed.	891027
J0014	HEALTH_AND_WELLBEING_SPECIAL_TES	HEALTH AND WELLBEING request for a special test resulting from a problem during the Health and Wellbeing test.	891027
K0052	INFLIGHT_HANDOFF	Sent from 2.3 SITUATION ANALYSIS to 2.2.2 MANAGE IN-FLIGHT to indicate when a plane is not the respons. of FLGHT ARRVLS	891030
H0027	INITIAL_SIMULATION_DATA		891025
K0028	INQUIRY_RESPONSE_CONTROL LOG_DAT	Data logged by 2.3.5 RESPOND TO INQUIRIES. Includes PILOT request, and the response to the request.	891027
E0007	INVALID_ATC_TRANS	ATC Transactions rejected for update by UPDATE REFERENCE DATA	891026
E0006	INVALID_GAA_TRANSACTIONS	GAA Transactions not accepted for update by UPDATE REFERENCE DATA.	891025
J0008	LOGGING_DECISION	Information from the AC_AND_ENV_DATA indicating the types of log data to be maintained in the ATC_LOG.	891027
J0006	LOG_DATA	The processed log data stored in the ATC_LOG.	891027
J0007	LOG_DECISION	A decision determined by LOG DECIDER indicating whether the log data of an identified type should be logged.	891027
H0005	MONITORING_LOG_DATA	Data logged by the MONITORING function	891027
H0014	NON_CLOSED_FLIGHT_PLAN_MESSAGE	Message generated to the external Search and Rescue entity when a plane is overdue according to its flight plan.	891027
K0053	PILOT_PROBLEM	Problem reported by the pilot, such as aircraft emergencies, etc.	891030
K0054	PILOT_REQUESTS	Request from the PILOT. Includes requests such as request for reroute, or information about weather, traffic, etc.	891030
K0030	PILOT_REQUEST_FOR_REROUTE	Sent from 2.3.5 RESPOND TO INQUIRIES to 2.2.2.1 INFLIGHT REROUTING when the PILOT requests a reroute.	891030
H0002	PILOT_WEATHER_REPORTS	Any weather data reported by pilots	891027
H0016	PREDICTED_FLIGHT_DATA	Predictions about the future path of identified aircraft while in flight. Used to predict/prevent collisions.	891027
J0004	REFERENCE_LOG_DATA		891026

Name	Alternate Name	Short Descrip.	Last Modify Date
E0005	REFERENCE_UPDATE_LOG_DATA	Log messages of REFERENCE UPDATE tables to be recorded by LOGGING.	891026
K0047	RELATIVE_POSITIONS	Distance (long, lat, and altitude) between an aircraft and other aircraft, weather, and terrain obstacles.	891030
K0039	REQUEST_FOR_GROUND_CLEARANCE	Request for GROUND CLEARANCE for flight indicated by FLIGHT PLAN ID.	891030
K0042	REQUEST_FOR_GROUND_TRAVEL_INST	Request for ground travel plan on arrivals.	891027
K0034	REQUEST_FOR_PUSHOFF/TAXI	Request from the PILOT to pushoff from the gate and taxi to the takeoff runway.	891027
K0055	REQUEST_LANDING_INSTRUCTIONS	PILOT request to 2.2.3 MANAGE FLIGHT ARRIVALS for landing instructions, including which runway.	891030
K0014	REROUTING CONTROL LOG DATA	Data generated by 2.2.2.1 INFLIGHT REROUTING to be logged. Includes new routes and updated flight plans.	891027
K0035	RESPONSE_FOR_PUSHOFF/TAXI	Response from 2.1.1 DEPARTURE GROUND TRAFFIC PROCESSING to the PILOT to his request to pushoff/taxi.	891027
K0013	RULES_AND_REGS		891025
K0015	SEQUENCING_CONTROL_LOG_DATA	Data generated by 2.2.2.2 INFLIGHT SEQUENCING to be logged.	891027
H0012	SIMULATION SURVEILLANCE RETURNS	Simulated surv. returns (such as RADAR and weather) used by MONITORING and CONTROL.	891027
H0029	SIMULATION_LOG_DATA	Data that needs to be written to the ATC system log.	891027
H0009	SIMULATION_SIGNAL	Boolean signal telling CONTROL and MONITORING the the system is in simulation mode.	891122
H0001	SITE_WEATHER_SURVEILLANCE_DATA	Weather data measured at a particular site-- today includes radar data and specific measurements(e.g., barometric press)	891027
K0026	SPACING_CONTROL_LOG_DATA	Data generated by 2.2.2.3 INFLIGHT SPACING to be logged.	891027
H0032	SPECIAL_INTEREST_REPORTS	Reports provided to the government regarding any events of special interest (e.g., presidential entcurage).	891027
H0031	SPECIAL_INTEREST_REPORT_REQUEST	Request by government for report of interest to government	891027
J0018	SPECIAL_TEST_ACK	An acknowledgement from MONITORING or CONTROL that the special test was performed.Test results are included in ack.	891027
J0016	SPECIAL_TEST_INITIATED	A signal to EVALUATE TEST RESULTS from INITIATE TEST to indicate that a test request was issued.	891027

Name	Alternate Name	Short Descrip.	Last Modify Date
J0017	SPECIAL_TEST_REQUEST	A request to MONITORING and CONTROL that a special test be performed.	891027
J0013	SPECIAL_TEST_REQUEST_FROM_LOGGIN	EXAMINE LOG DATA request for a special test resulting from an anomaly detected in the ATC LOG data.	891027
J0049	STATUS_EVENT_REPORTS	ATC status reports generated from the log data.	891027
H0018	STRIPS_(PLANNED/NONPLANNED_FLTS)	Abbreviated flight plan data for use by the CONTROL function in spacing and sequencing functions.	891027
K0006	SURVEILLANCE_DATA	Data passed from MONITORING to CONTROL which contains information about the environment (weather and traffic).	891027
K0044	TAKEOFF_CLEARANCE_REQUEST	Request from the PILOT to 2.2.1 MANAGE FLIGHT TAKEOFF to takeoff. Plane is at the end of the runway.	891030
K0045	TAKEOFF_CLEARANCE_RESPONSE	Response to TAKEOFF CLEARANCE REQUEST from 2.2.1 MANAGE FLIGHT TAKEOFF to PILOT. Allows PILOT to takeoff.	891030
H0017	TARGET_IMAGES	Identified aircraft in the airspace (e.g. radar blips)	891027
H0011	TENTATIVE_FLIGHT_PLAN	Data about a flight that does not have an associated flight plan.	891027
H0020	TERRAIN_SURVEILLANCE_DATA	Raw surveillance data about the terrain.	891027
H0034	TEST_ACK	Acknowledgement back to the TEST function at the completion of the requested test.	891106
J0015	TEST_INITIATED	A signal to EVALUATE TEST RESULTS from INITIATE TEST to indicate that a test request was issued.	891027
H0033	TEST_REQUEST	Request by the TEST function to perform a health/well being or special test.	891106
J0005	TEST_RESULTS	Data generated by TEST to be logged. It includes the results from the Health and Wellbeing and Special Tests.	891027
H0019	TRAFFIC_SURVEILLANCE_DATA	Raw surveillance data (e.g., radar) on aircraft.	891027
H0035	UPDATED_FLIGHT_PLAN DATA	Revised flight plan data for a particular flight which was revised automatically by the TRACKING function	891106
H0010	UPDATED_FLIGHT_PLANS	Updated flight plans as received from the CONTROL function, necessary due to rerouting and pilot requests.	891027
K0007	UPDATED_FLIGHT_PLANS	Updates for flight plans do to reroute. Sent from CONTROL to MONITORING and FLIGHT PLAN ENTERER.	891030
E0004	VALIDATED_REF_UPD_TRANSACTIONS	Validated transactions, to be processed by UPDATE REFERENCE DATA and stored in the appropriate data stores.	891025

Name	Alternate Name	Short Descrip.	Last Modify Date
K0056	WEATHER_ALERT	Using SURVEILLANCE DATA and REL POS, determines weather hazards such as strong winds or thunderstorms.	891027
K0016	WEATHER_ALERT_CONTROL_LOG_DATA	Data logged by 2.3.3 WEATHER AVOIDANCE. Included WEATHER ALERTS.	891027
H0003	WEATHER_REPORTS	Weather reports provided by an external agency, provides forecasted weather for example	891027
H0004	WEATHER_REPORT_REQUESTS	Request by a pilot for weather data.	891027
H0008	WEATHER_REPORT_RESPONSE	Response to the pilot to his request for weather data	891027